

Introducing OmniFind Analytics Edition: Customizing for Text Analytics



Installing and configuring



Customizing libraries and rules



Case studies



Wei-Dong Zhu
Srinivas Chitiveli
Kameron Cole
Scott Harms

Ratheesh Muraleedharan



International Technical Support Organization

**Introducing OmniFind Analytics Edition:
Customizing for Text Analytics**

June 2008

Archived

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (June 2008)

This edition applies to Version 8, Release 4, of IBM OmniFind Analytics Edition (Offering Identifier: P91230)

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks	x
Preface	xi
The team that wrote this book	xi
Become a published author	xiii
Comments welcome	xiv
Chapter 1. Introducing text mining and OmniFind Analytics Edition	1
1.1 Takmi	2
1.2 Text mining	3
1.2.1 Beyond search: Text mining versus search	4
1.3 Use cases for IBM OmniFind Analytics Edition	4
1.3.1 Find out what your customers really think	6
1.3.2 Discover problem trends: Find it before it breaks	7
1.3.3 Text mining and the law	8
1.4 Features and benefits of OmniFind Analytics Edition	8
1.4.1 Features of OmniFind Analytics Edition	9
1.4.2 Benefits of OmniFind Analytics Edition	11
Chapter 2. OmniFind Analytics Edition architecture	13
2.1 Product overview	14
2.2 System architecture	15
2.2.1 Data Ingestor	15
2.2.2 Natural language processing and OmniFind Analytics Edition Preprocessor	16
2.2.3 Indexer	18
2.2.4 Text Miner	18
2.3 Data processing flow	19
2.4 Topologies supported	20
2.5 Basic concepts	22
2.5.1 Category	22
2.5.2 Rules	23
2.6 Text analysis applications	23
2.6.1 Text Miner	23
2.6.2 Dictionary Editor	24
2.6.3 Rule Editor	25
2.6.4 Alerting System	25
2.6.5 Online manual	26

2.7 Integration with OmniFind Enterprise Edition	26
Chapter 3. Installation and configuration	29
3.1 Before you begin	30
3.1.1 Disk space and memory recommendation	30
3.1.2 System requirements	31
3.1.3 Installation file	32
3.1.4 Installation process overview	32
3.2 Installation and configuration for Windows platform	32
3.2.1 Installing OmniFind Analytics Edition on Windows	33
3.2.2 Configuring OmniFind Analytics Edition on Windows	38
3.3 Installation and configuration for AIX platform.	42
3.3.1 Installing OmniFind Analytics Edition on AIX	43
3.3.2 Configuring OmniFind Analytics Edition on AIX	48
3.4 Verification of installation and configuration	50
Chapter 4. Data ingestion and indexing	55
4.1 OmniFind Analytics Edition preprocessing overview.	56
4.2 Database directory creation and registration.	58
4.3 Preparation of CSV files	60
4.4 Database category creation	63
4.4.1 Editing the category tree (category_tree.xml)	65
4.4.2 Editing database settings (database_config.xml)	68
4.5 Generating ATML files.	70
4.5.1 Editing Data Ingestor configuration file (data_ingester_config_csv2atml.xml)	70
4.5.2 Invoke Data Ingestor command to convert CSV to ATML format	77
4.6 Natural language processing.	78
4.6.1 Allocating natural language processing resources	78
4.6.2 Generating MIML files.	79
4.7 Indexing.	81
4.7.1 Creating a new index	81
4.7.2 Updating an index by adding files	83
4.8 Accessing Text Miner application	83
4.9 Deleting data	84
4.10 Importing data using OmniFind Enterprise Edition	86
4.11 Quick reference for data ingestion and indexing.	86
Chapter 5. Text Miner mining basics	87
5.1 Text Miner	88
5.1.1 Deploying and accessing the application	88
5.1.2 Window layout and functional overview	89
5.1.3 Database selection window.	90

5.2 Category tree view	90
5.2.1 Sorting categories	92
5.2.2 Display and hide subcategories	93
5.2.3 Selecting categories and views	94
5.3 Search	96
5.3.1 Keyword search	97
5.3.2 Category search	101
5.3.3 Date search	104
5.3.4 Applying search operators	105
5.4 Views	111
5.4.1 Top view	111
5.4.2 Docs view	112
5.4.3 Category view	116
5.4.4 Time series view	119
5.4.5 Topic view	121
5.4.6 Delta view	125
5.4.7 2D Map view	128
5.5 Reporting features	132
5.5.1 Bookmark results	132
5.5.2 Reports	134
5.5.3 CSV output	138
Chapter 6. Text Miner advanced	141
6.1 Text analytics versus text mining	142
6.2 Frequency	142
6.3 Correlation	145
6.3.1 Correlation values in Category view: frequency versus correlation	145
6.3.2 Correlation values used in the 2D Map view	148
6.4 Topicality index	152
6.4.1 Topic view	153
6.5 Increase indicator	158
6.5.1 Increase indicator index in the Delta view	159
6.6 Conclusion: Understand what you see	162
Chapter 7. Customizing the dictionary	165
7.1 Dictionary Editor overview	166
7.1.1 Launching the Dictionary Editor	166
7.2 Working with categories and the category tree	167
7.2.1 Editing the category tree	168
7.2.2 Configuring the user interface	174
7.3 Working with keywords and synonyms	176
7.3.1 Editing keywords	178

Chapter 8. Customizing rules	189
8.1 Dictionary rules overview	190
8.2 The Rule Editor	192
8.3 Regular expression syntax	193
8.4 Creating a sample rule	196
8.4.1 Creating a POS constraint	197
8.4.2 Creating a string constraint	199
8.4.3 Setting the rule name and its value	201
8.4.4 Manually editing the rule for additional rule constraints	204
8.4.5 Converting the rule file from .rpf to .pat	205
8.4.6 Testing rules	206
8.5 Adding additional constraints	207
8.5.1 Adding a lex constraint	207
8.5.2 Adding a category constraint	211
8.5.3 Adding an ftrs constraint	214
8.6 Summary of rule creation	218
Chapter 9. Alerting System	221
9.1 Alerting System functional overview	222
9.2 Setting up alerts	222
9.2.1 Setting up Increase Detection	223
9.2.2 Setting up Correlation Detection	230
9.3 Batch processing and result analysis	235
9.3.1 Batch processing	235
9.3.2 Increase Detection results	236
9.3.3 Correlation Detection results	239
Chapter 10. Case studies	241
10.1 Case studies overview	242
10.2 Case study: Technical Help Desk	243
10.2.1 Overview of the business scenario	244
10.2.2 Technical Help Desk database details	244
10.2.3 Setting up the case study	247
10.2.4 Basic analysis	254
10.2.5 Root cause analysis	263
10.3 Case study: NHTSA	267
10.3.1 Overview of the business scenario	267
10.3.2 NHTSA database details	268
10.3.3 Basic analysis	270
10.3.4 Root cause analysis	273
10.3.5 Sample view usage	276
10.4 Case study: e-mail database	280
10.4.1 Overview of the business scenario	280

10.4.2 e-mail database details	281
10.4.3 Basic analysis	282
10.4.4 Sample rules	287
10.5 Apply new rules and dictionaries	291
Chapter 11. Integrating OmniFind Analytics Edition with OmniFind Enterprise Edition	295
11.1 Introduction to IBM OmniFind Enterprise Edition	296
11.2 Integration of OAE and OEE	297
11.3 Uploading OmniFind Analytics Edition annotator	299
11.3.1 Uploading and associating UIMA annotators	300
11.3.2 Start crawlers and parse	303
11.4 Building analytics index	304
11.5 Registering and browsing analytics index	306
11.6 Integrating custom dictionaries	309
11.6.1 Creating a custom dictionary	309
11.6.2 Registering OAE LanguageWare dictionary with OEE	311
11.6.3 Rebuilding the OmniFind Analytics Edition index	314
Related publications	317
Online resources	317
How to get Redbooks	317
Help from IBM	317
Index	319

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	LanguageWare®	Redbooks (logo)  ®
AIX 5L™	Lotus®	Tivoli®
Alerts®	Lotus Notes®	WebSphere®
DB2®	NetView®	Workplace™
DB2 Universal Database™	Notes®	Workplace Web Content Management™
FileNet®	OmniFind™	z/OS®
IBM®	QuickPlace®	
Informix®	Redbooks®	

The following terms are trademarks of other companies:

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

FileNet, and the FileNet logo are registered trademarks of FileNet Corporation in the United States, other countries or both.

AMD, the AMD Arrow logo, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

J2EE, Java, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Excel, Internet Explorer, Microsoft, SharePoint, SQL Server, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® OmniFind™ Analytics Edition uses information in unstructured and structured content to improve decision making. It provides unstructured information analysis, which can improve customer service levels, create cross-sell capabilities, and provide early problem detection.

IBM OmniFind Analytics gives business users the tools to gain valuable insights, facilitating better business decisions, by exploring and using key information assets in the enterprise. It also provides multiple ways to explore and analyze information, and it delivers sophisticated entity extraction capabilities when working with unstructured content. It is based on the open Unstructured Information Management Architecture (UIMA) standard.

This IBM Redbooks® publication will help you understand the power and capability of IBM OmniFind Analytics Edition Version 8.4. The book provides instructions on installing and configuring IBM OmniFind Analytics Edition. It explains how to use it for text mining, and how to work with Dictionary Editor and Rule Editor to customize the application for better analysis and discovery. The book also discusses how to set up the Alerting System to automatically watch for increased unusual activities. The integration with IBM OmniFind Enterprise Edition (OEE) is also covered.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, San Jose Center.

Wei-Dong Zhu (Jackie) is an Enterprise Content Management, Risk and Discovery Project Leader with ITSO in San Jose, California. She has more than 10 years of software development experience in accounting, image workflow processing, and digital media distribution. Jackie holds a Master of Science degree in Computer Science from the University of Southern California. Jackie joined IBM in 1996. She has managed the production of many Enterprise Content Management, Risk, and Discovery redbooks.

Srinivas Chitivel (Varma) is a lead developer on a product called OmniFind enterprise edition. He has been working at IBM for the past 9 years on technologies related to digital certificates, enterprise search and analytics. On many occasions Varma has executed the role of a customer advocate, where he

would architect and integrate IBM technology to address searching of enterprise content. In his spare time, he contributes articles for IBM developer works with solutions that describe better integration of IBM search technologies. He holds a Bachelors degree from Osmania University (India) and is certified by SUN as a Web and Business components developer.

Kameron Cole is a Complex Solutions Architect for the Center of Excellence, Content Management and Discovery. He began Java™ programming in 1996, developing mostly enterprise CORBA/Java solutions. He has a Master of Arts degree in Linguistics, with a focus on relational/generative syntax. He was invited to work in Budapest Hungary in 1989 for the Montana Linguistics Consortium, under the auspices of the Soros Foundation. He also has a Bachelor of Arts degree in Computer Science, specializing in compiler construction and natural language processing. Kameron is a Sun-certified J2EE™ Developer, IBM-certified WebSphere® Administrator, IBM-certified WebSphere Portal Administrator/Developer, and IBM-certified Enterprise Developer. He is co-author of Developing J2EE Applications with WebSphere Studio: IBM Certified Enterprise Developer (MC Press), as well as several IBM Developer Works cookbooks for OmniFind Enterprise Edition. Recently, he was part of the team for the first implementation of OmniFind Analytics Edition in the United States.

Scott Harms is an Enterprise Content Management, Senior IT Specialist based in San Francisco, California. He has more than 10 years of software support and technical sales experience in architecting solutions for content, process, records and digital asset management. Scott has worked in technical management roles supporting IBM's Informix® Software group before joining IBM's FileNet® Software group working with Manufacturing, Retail and Technology customers. He has held customer facing positions within multiple product areas for the IBM Information Management Software Division over a 7-year IBM career. Scott attended the University of Kansas with an emphasis in Civil Engineering studies. Scott is a certified Informix DBA as well as a certified FileNet P8 Administrator, and he continues to lead the development of Enterprise Content Management solutions for IBM customers worldwide.

Ratheesh Muraleedharan is an Enterprise Content Management Specialist with ISL in Bangalore, India. He has more than 3 years of experience in developing Enterprise Content Management solutions, Enterprise Search and text analytics. He holds a Bachelor degree in Electronics and communication engineering from Mahatma Gandhi University, India. Ratheesh joined IBM in 2005. His key skills are in OmniFind Enterprise edition, FileNet, and other Web content management technologies. On many occasions, he has been working with the partners and customers in architecting and implementing Enterprise Search solutions.

Thanks to the following people for their contributions to this project:

Yvonne Lyon
Deanna Polm
Emma Jacobs
International Technical Support Organization, San Jose Center

Tetsuya Nasukawa
Iwao Inagaki
Shin Takakura
Takuma Murakami
Shin Nakagawa
Hiroki Oya
Daisuke Takuma
Goh Tanaka
Yuta Tsuboi
IBM Yamato Software Development Lab, Japan

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- ▶ Send your comments in an e-mail to:

redbooks@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Introducing text mining and OmniFind Analytics Edition

In this chapter, we introduce the term Takmi, the concept of text mining and its differentiation from traditional search. In addition, we introduce IBM OmniFind Analytics Edition (OAE), its main features and benefits, and some typical use cases.

We cover the following topics:

- ▶ Takmi
- ▶ Text mining
- ▶ Use cases for IBM OmniFind Analytics Edition
 - Find out what your customers really think
 - Discover problem trends: Find it before it breaks
 - Text mining and the law
- ▶ Features and benefits of OmniFind Analytics Edition

1.1 Takmi

The word *takmi*, as shown in Figure 1-1, means skillful, or clever.



Figure 1-1 “Takmi” means “clever”

It can also mean “craftsman”, or “artisan”, as in Figure 1-2.



Figure 1-2 “Takmi” also means “craftsman”

In the modern enterprise, you must be clever to survive. The skillful enterprise will see into the future and thereby avoid missteps.

One of the intentions of the original creators of the Takmi software (presently known as OmniFind Analytics Edition) was to invent something that can do what human beings *cannot do*, in contrast to competing products, which try to *simulate* human behaviors.

Another goal was to go beyond just looking at large data from different points of view. The team of inventors wanted to develop a technology that can identify important issues, by truly mining the data and uncovering trends that would otherwise go undiscovered. An analogy might be a comparison of traditional data mining software to a CT scan, which provides cross-section pictures of the body. However, the CT scan cannot find cancer, and it cannot discover any health anomalies. In fact, it is not the purpose of the CT scan to find anything, It is the physician, the highly trained eye, which does the discovery.

In this chapter, we try to understand the concepts of text mining and text analytics, which form the basis for OmniFind Analytics Edition (OAE). Then, we look at three common scenarios where text analytics can be applied. We show you how OmniFind Analytics Edition, is the true “artisan,” in empowering business in these scenarios. Finally, we give you a short and to-the-point summary of OmniFind Analytics Edition’s features and what these features can do for your enterprise.

1.2 Text mining

OmniFind Analytics Edition is based on the concept of *text mining*, or *text analytics*. First, by text, we mean unstructured textual data. Think of data in a database as structured. It is organized into columns and rows whose titles tell us the type of content that the rows and columns contain. A text file, in contrast, is just letters and punctuation. You have no idea of what the text is about, unless you read it.

Next, we look at the concept of mining, as in gold mining. Mining for gold is a difficult process of digging and blasting through layers of rock and dirt to discover (or, sometimes *not* discover) something very valuable, namely, gold. Likewise, terabytes of textual data can be difficult to blast through (because it is unstructured), especially if you want to find something of value. The first dilemma is deciding what is valuable, and the second dilemma is teaching your software to understand and then discover that which you consider valuable.

At the lowest level, we must teach our software to make the same elemental distinctions about the world that we humans do. For example, the word *crash* typically conjures up a very disturbing notion in our minds. However, although disturbing, the notion might not be specific enough to conjure up an image in our mind. Another example is the word *car*; we see the word and a part of our brain associates that word with a picture in our minds of a thing. You will probably see a different thing in your mind, a different make of car, than the person next to you. The computer, on the other hand, just sees three letters.

Through a process referred to as *natural language processing*, software can recognize strings of letters such as things and actions and tag them with their parts of speech accordingly. Thus, crash is tagged as a verb and car is tagged as a noun. So, we have begun to mine, but we have not struck gold yet.

These two words begin to have value to us when we establish a relationship between them; in other words, when we see the words *car crash* together, we immediately get a fairly concrete image of broken glass flying and metal being crumpled. But it is not the individual words that give us this picture, nor is it the

part of speech. Rather, it is a higher level relationship between the two notions that gives us the picture.

So, our software needs to catalog these kinds of relationships and then apply them correctly in order for mining to take place.

Text analysis comes at an even higher level of relationship: that of *correlation*. If a software program can discover from a particular dataset that, in the United States, car crashes are responsible for 100 deaths a day, then the software has taught us something. The software has analyzed the data and thereby given us a much more meaningful response, a conclusion, a cause for 100 deaths per day. In the end, the software leads us to conclude that car crashes are the United States' leading health problem. This is valuable information. It is more valuable than if we perform a traditional search of the same data.

1.2.1 Beyond search: Text mining versus search

So then, what is the difference between text mining and search? Is there a difference? Certainly, there are search engines that search unstructured data. But they do not *mine*; they do not *analyze*. It is not the goal of a search engine, typically, to perform these tasks. A search engine's job is to reduce the size of a large dataset to something more manageable, by separating relevant documents from irrelevant documents, based on search criteria, and to rank the relevance of these documents.

Text mining has a scientific goal: to discover strong generalizations in a dataset. While these generalizations can be used to further reduce the size of the dataset, this is not the goal of the mining. A strong generalization is, in science, a very valuable thing.

It is important to stress that text mining does, indeed, search, just as the search engine does. The point is that text mining draws meaningful insights from the search results *in addition to searching*.

1.3 Use cases for IBM OmniFind Analytics Edition

It is safe to say that text analytics has not become an industry buzz word quite yet. This is due in part to the fact that the concept is fairly complex and not in the domain of common human knowledge, like search is. However, text analytics is, in fact, no less than the next major step in the evolution of electronic information.

The history of the Internet, as it is recognized and used by much of the world, is well-known: Web content began as static HTML files on Web servers. But soon thereafter, it became fairly commonplace to dynamically read data from databases and transform it into Web content. This added significantly to the overall content on the Web: Whatever content contributors add in terms of static content, a time-consuming task, was augmented by data from relational databases, which was already written. Meanwhile, traditional content from books and journals was being transferred by a variety of manual means.

Some of this content was transferred carefully, with an eye to its eventually being searched. The content was given structure, in the form of meta-languages such as XML. XML, like columns and rows in a database, provides *metadata*, to say what the content is about. Much of the traditional content, however, was simply copied or scanned into text or HTML files, with no structure.

Within recent years, the move has been to putting author content directly on the Web. There is content-authoring software — for example, software for the creation of blogs and wikis. This content, like the original static Web, is also mostly unstructured.

Finally, the increasing amount of electronic data is contributed by e-mail and instant messaging, both unstructured, as well. The net effect is exponential growth in total electronic content, with the unstructured content greatly outweighing the structured.

The public popularity of Internet search engines in recent years reflects the fact that our society is aware of the value of the electronic content now available and the need to somehow *manage* this content. Managing the content is more than just searching, in the same way as managing the flow of vehicular traffic prevents it from becoming simply chaotic and dysfunctional. There is a growing dissatisfaction with search engines, because the public, growing ever more search savvy, realizes that the search results that they are getting are not the ones they want or expect.

The next step, then, is for content searchers, both public and private, to understand why this is the case: the search software available to most users does not have the ability to “read” the content that it is searching. This would require structure, which, in the case of all the unstructured content, would somehow mean adding metadata. Further, despite all search result ranking algorithms, the only way to return an intuitively satisfying search result set is to add linguistic and statistical analysis of the metadata to the capabilities of the search software. This is the foundation for OmniFind Analytics Edition.

To better understand the critical significance of text analytic software, we look at real-world use case scenarios where OmniFind Analytics Edition has provided the ultimate solution to business needs.

1.3.1 Find out what your customers really think

Do you hear the voice of your customers? In the Information Age, the voice of customers is readily available. The voice of customers is *information*, just like the Library of Congress online catalog or the archives of your favorite news groups. In fact, the voice of customers is more available today than it has ever been in history, precisely because the enterprise Web technology has provided so many ways for customers to express their opinions.

There are few serious businesses today that do not provide support by e-mail. Savvy companies also provide their customers with forums and list-serve groups. Outside the enterprise, there is the vast world of blogs, instant messaging, and user groups overflowing with opinions, support information, rants and raves about every and any product or service you can imagine — but far beyond the control and oversight of those product or service providers. These sources of customer opinion are far more reliable than traditional surveys — self filtering factors in customer surveying make it largely unreliable as a true source of insight. On the contrary, people tend to be freer in expressing their opinions than they might be in their normal, face-to-face interactions with others, simply because on the Internet, they are faceless.

Clearly, all this information must be tapped. Its value to the enterprise, either beneficial or detrimental, should be obvious. However, the problem remains of how to collect the information, and, more important, how to distill its message. It is unstructured information. There are companies who pay rooms full of workers to manually scour millions of E-mail messages, online product reviews, call-center notes, survey verbatim, customer-relationship management (CRM) text fields, press releases, and other textual information available across and beyond an enterprise.

Electronic search can find negative words, phrases, across such an expanse of data; however, even the most complete search collection would have not much value without mining, without analysis. Consider the variety of ways in which people express their likes, dislikes, not to mention countless other gradations of sentiment. It is simply not possible to use search to discover sentiment, because it is anathema to the nature of search: search finds what you are looking for, because you tell it what you are looking for, because you *know* what you are looking for. Discovering your customers' opinions is looking for *unknowns*.

Imagine the application of text mining in a product launch: a very wide-reaching public consensus to be determined, by analyzing blogs and forums in the product space, *prior* to the launch, which would then inform an iterative, effective marketing strategy. Further, mining would provide insight into why the product was eagerly anticipated, or dully ignored, and appropriate adjustments could be made in both the product and the product message to electrify the positive and cull the negative.

Finally, there is great value in software that can integrate unstructured customer data, with structured data — the traditional measures of product success such as transactions, sales volume, and consumer spending indexes. By normalizing customer sentiment with quantifiable sales trends on a date range, useful correlations could be made in an empirical way. For this reason, OmniFind Analytics Edition combines both structured and unstructured data for analysis.

1.3.2 Discover problem trends: Find it before it breaks

The fundamental nature of support is fixing problems. The enterprise employs countless employees to read the incoming deluge of questions, complaints and problems. The expected response of the support employee is to get the problem fixed. They are not expected to determine the cause of the problem, nor are they expected, or even capable of, discovering long term, over-arching trends.

However, it is the recognition of these trends that allows companies to determine the need to issue recalls, for example. Early detection of a problematic behavior, which could lead to catastrophic failure, is clearly something the agile enterprise cannot do without. Finally, although somewhat less dramatic, consider the ability to reduce noise. For example, e-mails from solicitors and network system messages can be eliminated from the result set, making the in-depth analysis of relevant data infinitely more manageable. The solution lies in text analytics.

Imagine the frequency, in a support call center, of a word such as upset — people are often upset when they get to the point of calling in. Without analysis, our call center might just take it for granted that callers would be upset, and not investigate further. OmniFind Analytics Edition, in contrast, will use its correlation feature to display immediately why the callers are upset — the highest causes for a customers' being upset. To our surprise, it might be something entirely different than frustration with the product: It might be dissatisfaction with the quality of service of your call center itself, for example, due to the perceived lack of knowledge and experience of the call center agent. This is something a skillful company will want to know as early as possible, and fix.

Correlation is often surprising to us humans. These surprises can be dangerous. It is, in fact, the surprising correlations that OmniFind Analytics Edition want to bring to light. It would not be a surprise to see that an automobile's ignition caused engine fires. It would be a surprise to find out that the fires were actually caused by the cruise-control switch. Search, without correlation, would never surface a problem with the cruise-control. Even deeper: while we might *expect* a high correlation between the terms "airbag deploy" and "accident," OmniFind Analytics Edition can take us deeper into this correlation, and show us that the airbag deployment was the *cause* of the accident, not, as expected, the desired *result*.

Finally, analytics can teach you how to better organize your call center. You might discover that calls are being routed incorrectly — your PC help desk is wasting considerable time re-directing callers with telephony issues. Perhaps your prompts list for PC help needs a prompt for phone help, or, the phone help number is not visible enough to customers. None of these trends would be recognized as quickly, if at all, by your call agents. Remember, OmniFind Analytics Edition was designed to do something humans cannot do.

1.3.3 Text mining and the law

Daily business proceedings have always been somehow recorded, whether by the stack of transaction receipts lying on the clerk's desk at the end of the business day in the 1800's, or the audit log for database writes in the 21st century.

Within just the past few years, this large, but nebulous record of business has become much richer due to the wide adoption of e-mail and messaging by the corporate world. Imagine how much greater detail is contained in an instant messaging exchange between, say, two government workers, chatting about a particular case, compared to ten years ago, when the exchange might have been limited to a phone call or two, a faxed request/response.

Who in the corporate world today has not experienced the e-mail chain syndrome? The entire evolution of the solution to a problem, or the creation of a problem, is recorded in a string of forwards. Think of the corporate litigations that the average person is aware of, within recent years, that have depended critically on the contents of such e-mail chains.

The choice is clear: The enterprise must scrutinize this information. The solution is also clear: Text analytics offers a way to discover malicious or compromising messages, at this volume of data flow. The legal community has recognized the need for software that does precisely what OmniFind Analytics Edition does.

1.4 Features and benefits of OmniFind Analytics Edition

OmniFind Analytics Edition makes complex analytical tools available to a wide range of customers, where before this complexity had been often too challenging. With OmniFind Analytics Edition, these tools make complex statistical and linguistic analysis transparent to the user, without reducing their power or efficacy. In this section, we summarize the technology as a whole, and present its benefits.

1.4.1 Features of OmniFind Analytics Edition

These are the features that enable *actionable insight* offered by OmniFind Analytics Edition:

- ▶ Keyword search:

A keyword is combination of a character string *and* a category. This is quite different than search engine technology, which searches on character strings only. It is also different than search engines that incorporate some degree of categorization. OmniFind Analytics Edition considers grammatical units as keywords, not just individual words. Thus, “software...uninstall” is its own keyword.

- ▶ Merge unstructured data with structured data for deeper analysis:

OmniFind Analytics Edition imparts structure to unstructured data, according to the users’ designs. The new structure of the once-unstructured data can be altered to synchronize with the unchangeable structure of the permanently structured data. The combination of these two datasets can uncover new trends that would be otherwise undiscovered.

- ▶ Sophisticated semantic search:

Semantic search is where, instead of entering a keyword, the user enters a phrase. OmniFind Analytics Edition recognizes that this search entry is, indeed, a grammatical, semantic unit — not just a collection of search terms. Furthermore, OmniFind Analytics Edition disambiguates search terms by understanding the context. Thus, the word “bark” in “tree bark” is not the same as “dog bark.” For its semantic search, OmniFind Analytics Edition uses the open source platform Unstructured Information Management Architecture (UIMA), developed by IBM Research, currently incubating at the Apache Software Foundation: <http://incubator.apache.org/uima>. We discuss UIMA in detail later in this book.

- ▶ Drill-down navigation:

Analytic search in OmniFind Analytics Edition is a learning process. There is no particular document that the user is looking for. Rather, the user wants a collection of relevant documents returned that will answer a larger question. While this collection can answer the user’s question via statistical analysis, as in trend analysis and delta analysis (discussed below), sometimes the answer is in the documents themselves. OmniFind Analytics Edition lets you follow your query all the way down to viewing the entire document in the Text Miner application. This contrasts typical search results, which only give a summary. If the users need to view the original documents, they click on a link, which in turn attempts to retrieve the original and invoke the appropriate viewer for the file type — a process that more often than not fails. OmniFind Analytics Edition’s increases usability of search by maintaining full copies of all documents, for viewing and analysis.

► Trend analysis:

OmniFind Analytics Edition can extract trends from data corpora in several ways. First, the search results can be displayed via distribution over time. In other words, the query “sell stock” can be shown as a general user trend over time, expressed serially, in a linear graph, perhaps revealing a spike in selling stock prior to a major product release, for example. This is done in the Time Series view. Alternatively, the same query can be viewed in a cellular matrix, which reflects deviation of the frequency of the query (keyword or subcategory) per date. This produces a kind of heat map view, referred to as the Topic view, in the Text Miner application. Regardless of the view, the point is that OmniFind Analytics Edition delivers insights to data, not just the data itself.

► Delta analysis:

The delta analysis feature of OmniFind Analytics is a true attempt to do something that humans cannot do (or do it easily in a timely fashion) — to see into the future. Data is viewed as a function of change in frequency over time. Based on the rate and direction (up or down) of the change, future trends are projected in the graphical chart, and heat indexes can be applied, to alert the user of a trend which is contra-indicated.

► Correlation:

Correlation is shown between the search terms entered and any/all keywords and subcategories in the Category view. This means that *any search* is automatically part of a statistical analysis of correlation strength — a feature that distances OmniFind Analytics Edition from any competitor. A 2D Map view shows the correlation between a vertical category — this could be a term, a concept, a phrase, for example — and a horizontal category. Together, these views give the user intuitive images of the data corpus. Of course, complex statistical methods for achieving reliability, over small datasets, and, taking into account noise, have been carefully implemented. Thus, the user involved in heuristic analysis is always presented with a highly reliable image of the prominence of terms or concepts in the overall dataset.

1.4.2 Benefits of OmniFind Analytics Edition

These are the benefits of *actionable insight* offered by OmniFind Analytics Edition:

- ▶ Improve customer satisfaction by resolving cases more accurately and in a more timely fashion.
- ▶ Improve customer retention by discovering what makes them unhappy as early as possible, so that you can address it before they leave.
- ▶ Improve your product image by discovering problems early, and correcting them.
- ▶ Improve your understanding of your customer by learning to speak their language. Discover the words they use to talk about your product.
- ▶ Improve efficiency of customer care by providing customer-facing agents with the most accurate view of the customer, and by improving your self-service knowledge base in a timely and efficient manner.
- ▶ Improve effectiveness of marketing campaigns by analyzing potential customer bases and trends through blogs, lists, and message boards.
- ▶ Improve sales by providing sales people with contextually relevant cross-sale and up-sale prompts.
- ▶ Reduce risk and financial exposure by efficient compliance monitoring and surveillance.

The current wave of search is clearly semantic search, with text mining. Enterprises relying on traditional navigational or research search solutions will not only falter behind their competition in terms of customer awareness, but, in cases of product failure litigation or compliance litigation, might also be unable to meet the standards of the current state of eDiscovery.

It is thus unlikely that any enterprise, large or small, will be able to circumvent the adoption of some degree of text mining. However, their choice in software should be based on open industry standards already in existence. Moreover, as has been shown with J2EE, software that incorporates open source platforms, such as UIMA, far outperforms the proprietary competitor in meaningful (and remunerative!) ways. OmniFind Analytics Edition is the only software on the market currently to meet these criteria.

Archived

OmniFind Analytics Edition architecture

This chapter provides an overview of the architecture and processing flow of IBM OmniFind Analytics Edition (OAE). We describe the main functions and basic concepts that are necessary for understanding these functions, the basic building blocks of OmniFind Analytics Edition and their interaction with each other.

We cover the following topics:

- ▶ Product overview
- ▶ System architecture
- ▶ Data processing flow
- ▶ Topologies supported
- ▶ Basic concepts
- ▶ Text analysis applications
- ▶ Integration with OmniFind Enterprise Edition

2.1 Product overview

IBM OmniFind Analytics Edition (OAE) uses information in unstructured and structured content to improve decision making. It provides unstructured information analysis, which can improve customer service levels, create cross-sell capabilities, and provide early problem detection. IBM OmniFind Analytics Edition gives business users the tools to gain valuable insights, facilitating better business decisions, by exploring and using key information assets in the enterprise. It also provides multiple ways to explore and analyze information; and it delivers sophisticated entity extraction capabilities when working with unstructured content. It is based on the open Unstructured Information Management Architecture (UIMA) standard.

OmniFind Analytics Edition allows users to explore, integrate, and analyze information extracted from unstructured sources together with structured data. It uses natural language processing capabilities to process unstructured content to classify and extract pertinent information such as entities, relationships, sentiments. It then analyzes this information together with the related structured data to leverage all available information.

As mentioned in 1.4, “Features and benefits of OmniFind Analytics Edition” on page 8, OmniFind Analytics Edition provides the following features:

- ▶ Keyword search
- ▶ Merge unstructured data with structured data for deeper analysis
- ▶ Sophisticated semantic search
- ▶ Drill-down navigation
- ▶ Trend analysis
- ▶ Delta analysis
- ▶ Correlation

In addition, OmniFind Analytics Edition offers the following additional ways to explore and analyze information:

- ▶ Categorization
- ▶ Topic analysis
- ▶ 2D heat maps
- ▶ Automated alerting

2.2 System architecture

Key components of OmniFind Analytics Edition include:

- ▶ Data Ingester
- ▶ OmniFind Analytics Edition Preprocessor
- ▶ Indexer
- ▶ Text Miner

Figure 2-1 shows the OmniFind Analytics Edition system architecture with these components.

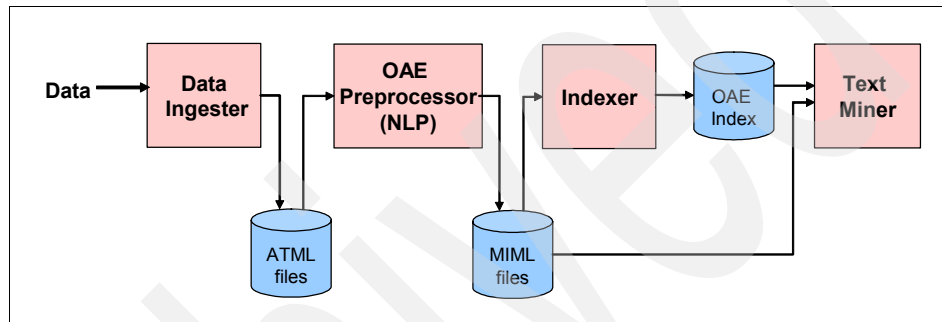


Figure 2-1 OmniFind Analytics Edition system architecture

These components can be broadly classified into two categories, information extractors or preprocessors, and information analyzers. Information extraction or preprocessing consists of processing raw data, converting it into the standard input format for OmniFind Analytics Edition (Data Ingester), and performing natural language processing of the input file. Information analyzers consists of indexing the data and analyze the data.

2.2.1 Data Ingester

Data Ingester is the component used to convert the data in Comma Separated Value (CSV) format into Advanced Text analytic Markup Language¹ (ATML), the standard input format for OmniFind Analytics Edition. The ATML files contain both structured and unstructured attributes for each document. The data to be analyzed, including text, should be prepared in CSV data format. Most of the relational databases and spreadsheet applications support exporting data to CSV format. The details about the different columns in CSV files and the data contained in these must be recorded in the hData Ingester configuration file for CSV to ATML conversion.

¹ See the definition of ATML in the shaded box.

There is a set of rules to which you have to adhere while creating the CSV files. For example, each CSV file should not contain more than 50000 lines, or the first line of the CSV file should be used to list column names and start describing data from the second line. These rules are discussed in detail in Chapter 4, “Data ingestion and indexing” on page 55.

2.2.2 Natural language processing and OmniFind Analytics Edition Preprocessor

Natural language processing (NLP) is the process of analyzing text and extracting entities of interest such as various parts of speech, entities related to a particular domain, and the relationships among the entities. It is built on the Unstructured Information Management Architecture (UIMA), an open software framework for enablement and collaboration around the creation, development, and deployment of technologies for discovering the latent meaning, relationships, and relevant facts buried within unstructured information.

Language processing is done for each input data (ATML file). The language processor analyzes and extracts the different entities contained in the input data using a chain of UIMA text analysis engines (TAE) to tokenize, parse, and apply dictionaries and rules. The output of this is stored as annotations in Mining Markup Language (MIML) files, which is the standard output format for OmniFind Analytics Edition preprocessor component, along with the original data.

Definitions of terms:

- ▶ *Advanced Text analytic Markup Language* (ATML) is an OmniFind Analytics Edition proprietary file format, and OmniFind Analytics Edition expects the target data to be in ATML format.
- ▶ *Mining Markup Language* (MIML) is another OmniFind Analytics Edition proprietary file format. A MIML file would contain the target content and extra metadata that gets discovered during the NLP processing. Natural Language Processing (NLP) is applied to the converted ATML content to MIML.

Figure 2-2 shows the internal architecture of OmniFind Analytics Edition natural language processor for English.

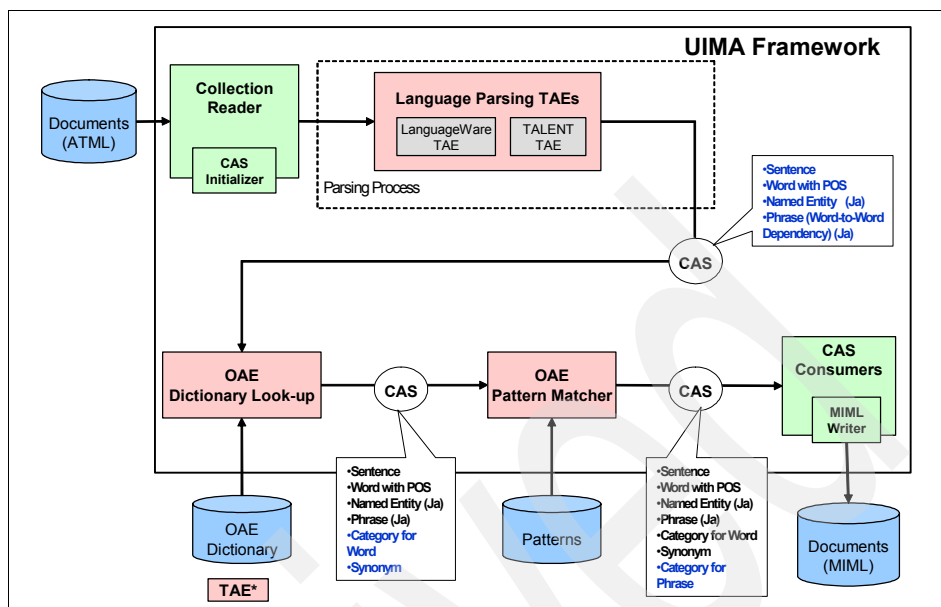


Figure 2-2 Internal architecture of OAE Natural language processor for English

Definition of terms:

The *Unstructured Information Management Architecture* (UIMA) framework is an open, industrial-strength, scalable, and extensible platform for building analytic applications or search solutions that process text or other unstructured information to find the latent meaning, relationships, and relevant facts buried within.

A *Common Analysis Structure* (CAS) defines how these annotators represent and share their results. The CAS is an object-based data structure that allows the representation of objects, properties, and values. The CAS logically (if not physically) contains the document being analyzed.

UIMA background information

An example UIMA application might ingest plain text files and identify entities, such as persons, places, and organizations; or relations, such as works-for or located-at. UIMA enables such an application to be decomposed into components, for example “language identification” to “language specific segmentation” to “entity detection (person/place names)”. Each component must implement interfaces defined by the framework and must provide self-describing metadata via XML descriptor files. The framework manages these components and the data flow between them.

The basic building blocks of UIMA are Analysis Engines. Analysis Engines (AEs) are components that analyze a document and infer and record descriptive attributes about the document as a whole, or about regions therein. This descriptive information, produced by AEs, is referred to generally as analysis results. UIMA provides a basic component type intended to house the core analysis algorithms running inside AEs. Instances of this component are called *Annotators*.

UIMA history

UIMA began as an IBM internal project and was developed by teams in IBM Research and IBM Software Group. For IBM, UIMA is a delivery vehicle of technology, as well as an enabling technology for academic and business partners. As more and more technology outside of IBM was built on top of UIMA, there was an increasing demand to open source UIMA. The first open source version of UIMA was released on SourceForge in early 2006. At the time, IBM did not accept external contributions to the source code. Later in the same year, in October 2006, the UIMA framework was accepted as an Incubator project at the Apache Software Foundation. This opened up UIMA development to a wider, non-IBM community.

2.2.3 Indexer

The *Indexer* component of OmniFind Analytics Edition indexes the processed MIML files. These indexes along with the MIML files are then fed to the Text Miner application. There are two types of indexing processes in OmniFind Analytics Edition, creation of new indexes and update of indexes by file addition. A new index is created for all the data processed by the language processor. When the index update happens, the indexer component checks the MIML files for data that was newly processed by natural language processing and creates an index for it. There are a number of sub processes which the indexer executes before the final index is created. First it updates the MIML file list in the `database_config.xml` for the particular database which is been indexed. Then it processes the individual MIML files and creates an intermediate index file for each MIML. These intermediate indexes are then merged to form the final index.

2.2.4 Text Miner

Text Miner is a Web-based application that dynamically performs statistical analysis of the results of language processing. It enables users to analyze the processed data by selecting categories, providing search conditions and further drilling down and analyzing data using different techniques and views such as trend analysis, delta analysis, Time Series view, Topic view, and 2D Heat Map view. It also provides us with useful insights, on the different trends and predominant patterns hidden in the dataset, with minimal analysis steps.

We also introduce Text Miner in the 2.6, “Text analysis applications” on page 23.

2.3 Data processing flow

Data processing flow within OmniFind Analytics Edition is as follows:

1. The Data Ingestor component extracts and transforms the input data in the CSV format into ATML files.

If the original data is not in CSV format, then the first step would be to convert the data into CSV format (using scripts, programs, or macros) and then having Data Ingestor convert it to ATML files. CSV is *the* input format that is supported by Data Ingestor.

2. The ATML files act as inputs for the NLP component. NLP analyzes and extracts entities such as parts of speech, named entities, and concepts, and creates the MIML files as output. The MIML files contain the results discovered by NLP along with the original data.
3. The MIML files are then indexed by the Indexer component to facilitate easy search, retrieval, and analysis.
4. These indexes along with the MIML files are fed to the Text Miner application for the users to perform functions such as multi-faceted search of the document collection, time series, delta analysis, and 2D correlations across the different categories.

Figure 2-3 illustrates the data processing flow of OmniFind Analytics Edition.

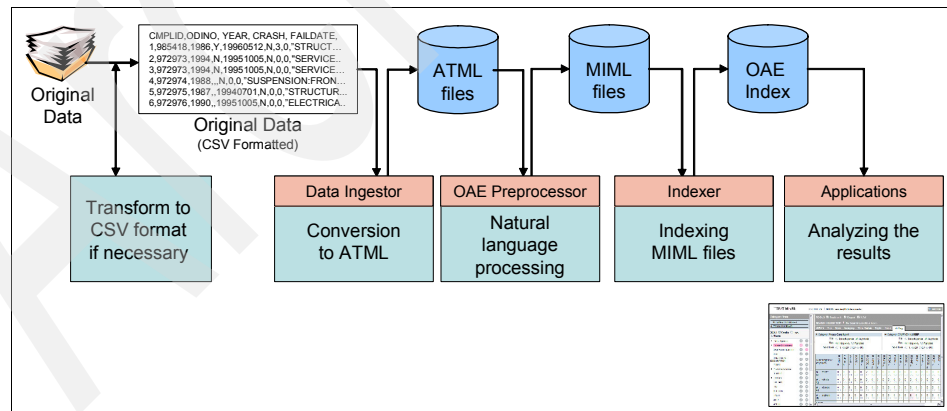


Figure 2-3 Data flow within OmniFind Analytics Edition

2.4 Topologies supported

OmniFind Analytics Edition supports three system topologies:

- ▶ Single server topology
- ▶ Two server topology
- ▶ Multi-server topology

Single server topology

A single server topology is one where all the components are on the same server. See Figure 2-4.

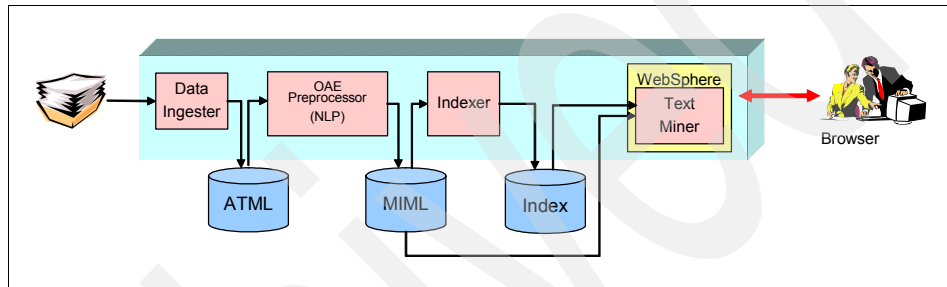


Figure 2-4 Single server topology

In a single server topology, the Data Ingestor, Natural Language Processor, Indexer components, and the application server hosting the OmniFind Analytics Edition applications (including Text Miner) are all reside on a single server.

Two server topology

A two server topology is one where the components are installed on two servers. See Figure 2-5.

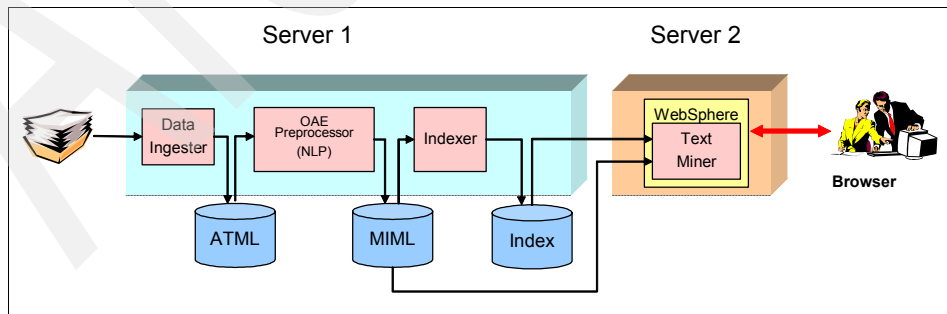


Figure 2-5 Two server topology

In a two server topology, the Data Ingestor, Natural Language Processor, and Indexer components are installed on the first server; the application server hosting the OmniFind Analytics Edition applications is installed on another server. The OmniFind Analytics Edition database with the MIML files and the Indexes on the first server have to be placed on a shared file system that is accessible to the second server.

Multi-server topology

A multi-server topology is one where the components are installed on three or more servers. See Figure 2-6.

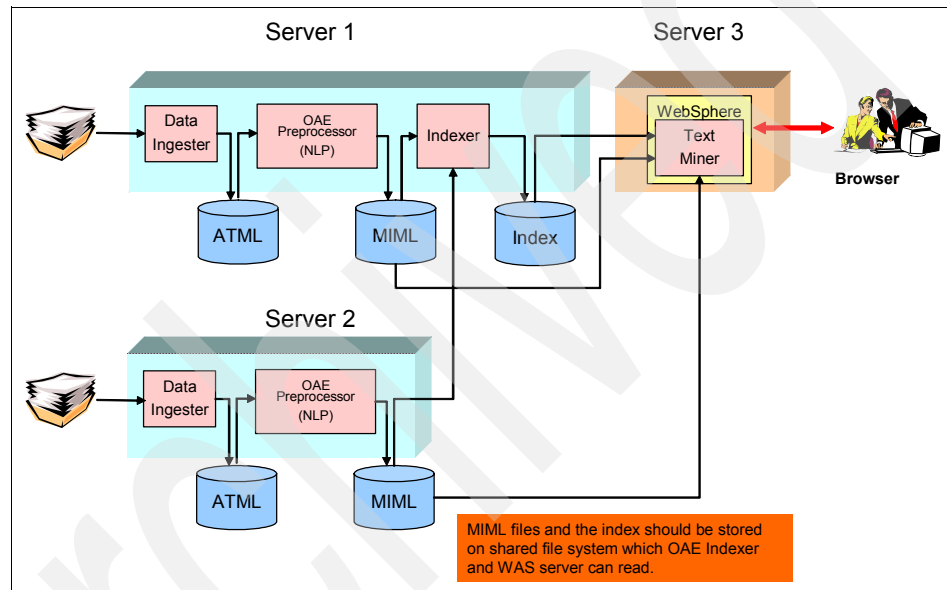


Figure 2-6 Multi-server topology

In a multi-server topology, multiple Data Ingesters and Natural Language Processors can reside on multiple servers, but there is only one single Indexer component. You can have multiple application servers hosting Text Miner and other OmniFind Analytics Edition applications.

All the MIML files created by the different language processing components should be stored in the MIML directory of OmniFind Analytics Edition database on a shared file system that is accessible to Indexer and the application servers. The indexes created from the MIML files are also stored in the index directory of OmniFind Analytics Edition database on a shared file system to facilitate easy access by the OmniFind Analytics Edition applications.

2.5 Basic concepts

In order to have an overview of OmniFind Analytics Edition, it is important to understand the basic concepts it uses. We use a sample database from National Highway Traffic Safety Administration (NHTSA) to provide examples for the concepts explained.

Database

OmniFind Analytics Edition manages the resources and results of language processing for each dataset (to be analyzed) in units called *databases*. The database in OmniFind Analytics Edition world, is not the same as the relational database.

2.5.1 Category

For every OmniFind Analytics Edition database, there are different tokens that can be associated with the documents that share similar patterns or content. The tokens help users analyzing the results of language processing better. *Category* is a label name given to the different tokens or keywords present in a data type.

Each data type can have multiple categories. For example, when we load the NHTSA dataset into OmniFind Analytics Edition, it creates the default categories corresponding to the different fields in the dataset such as fail date, model, and city. If a user wants to analyze the data based on the different components or parts of vehicles, then a new category for vehicle components can be created and data can be categorized based on the components involved.

The categories in OmniFind Analytics Edition are divided into:

- ▶ Standard item category
- ▶ System category
- ▶ User defined category

A *standard item category* is the field value that is attached to the raw data to be analyzed. An example for the standard item category are the date of inquiry and department, which are field values provided as part of the raw data. *System categories* are the default categories provided by OmniFind Analytics Edition, such as generic nouns and verbs. A *user defined category* is defined by users based on data characteristics. An example is the component that we mentioned in the previous paragraph.

We provide more explanation about categories, their creations, and usage in later chapters of the book.

2.5.2 Rules

The *rule* is the entity that allows the user to specify what is to be mined from the various documents.

A rule consists of two parts:

- ▶ A pattern
- ▶ A value

The *Rule Editor* helps the user define the pattern by enabling the configuration of a set of parameter values. The rule value specifies what the result of applying the pattern to the document should be. For example, we can create a rule for the NHTSA dataset that would detect all the documents related to negative events or troubles such as damage to a component, a crack in a component, or a tire blowout. To create this rule, we can list the different words or phrases that can be identified as a negative event or trouble in the pattern value field.

We provide more explanation about categories, their creations, and usage in later chapters of the book.

2.6 Text analysis applications

OmniFind Analytics Edition offers five Web based applications for text analysis:

- ▶ Text Miner
- ▶ Dictionary Editor
- ▶ Rule Editor
- ▶ Alerting System
- ▶ Online manual

In this section, we briefly introduce these applications. For more detailed usage of the applications, read the corresponding chapters of the book.

2.6.1 Text Miner

We mentioned Text Miner, one of the key components of OmniFind Analytics Edition in 2.2, “System architecture” on page 15. *Text Miner* is a Web-based application that does statistical analysis of the results of the language processing. It enables users to analyze the processed data by selecting categories, providing search conditions, and further drilling down and analyzing data using different techniques and views such as trend analysis, delta analysis, Time Series view, Topic view, and 2D Heat Map view. It also provides useful insights on the different trends and predominant patterns hidden in the dataset, with minimal analysis steps. Figure 2-7 shows the user interface of Text Miner.

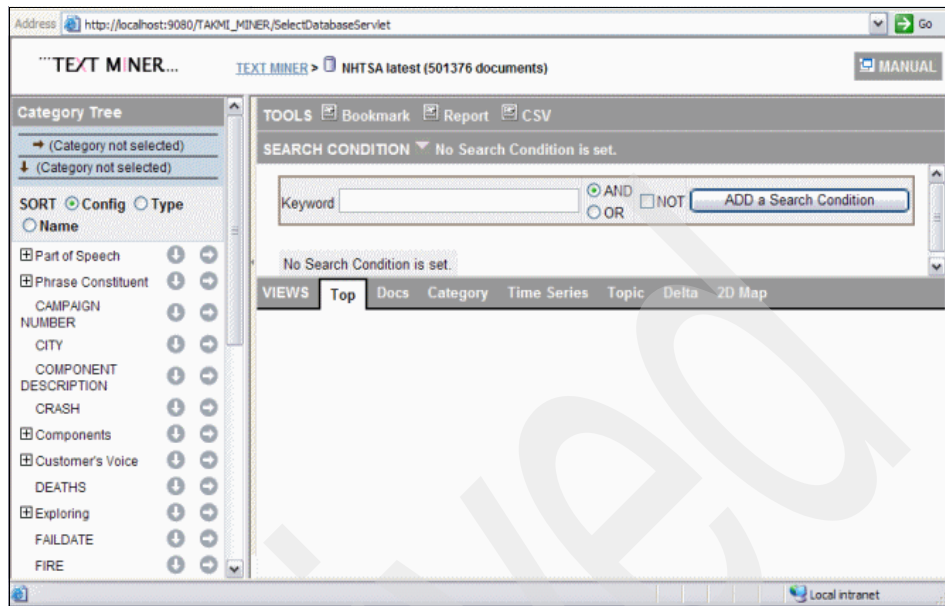


Figure 2-7 Text miner user interface

As shown in Figure 2-7, the Text Miner application interface consists of four areas: Category Tree, Tools, Search, and View. Users can select categories to be analyzed for a particular data type in the Category Tree area, and the Tools section offers the options of book marking and exporting the analysis results into HTML reports or CSV format. The Search and View sections enable users to further analyze and drill down on the data pertaining to different categories.

For more information on the Text Miner, read Chapter 5, “Text Miner mining basics” on page 87 and Chapter 6, “Text Miner advanced” on page 141.

2.6.2 Dictionary Editor

Dictionary Editor is a Web-based application that is used to edit categories, keywords, and synonyms. The main functions of the Dictionary Editor are to:

- ▶ Add or delete categories.
- ▶ Add or delete keywords or register them in a category.
- ▶ Add synonyms to keywords or delete synonyms from keywords.

The information about categories, keywords, and synonyms can be found in the dictionary resource files category tree file, keyword file, and candidate word file. The category tree file contains the entire category tree for a particular data type, and the keyword file saves a list of keywords and synonyms with their category information.

Usually keyword files are created for each category. A candidate word file loads frequently used character strings that have been extracted, or lists of product names or service names retrieved from internal databases, into the OmniFind Analytics Edition dictionary. The Dictionary Editor provides the Web based user interface to edit these resource files to create or edit categories, keywords, and synonyms for different data types and it also supports editing operations by multiple users.

For more information on the Dictionary Editor, read Chapter 7, “Customizing the dictionary” on page 165.

2.6.3 Rule Editor

Rule Editor is a Web-based application that is used to create and edit rules in the format supported by OmniFind Analytics Edition. These rules can be used subsequently by Text Miner for mining information from the document collections. The Rule Editor is linked with the dictionary tree, and supports operations on rules that are associated with a category in the category tree. The main functions of the Rule Editor are to:

- ▶ Add or delete rules under different categories of the dictionary tree.
- ▶ View and edit rules; change constraint values, reorder tokens, edit rule names or rule values.
- ▶ Search rules based on rule name or based on a constraint type and value.

For more information on the Rule Editor, read Chapter 8, “Customizing rules” on page 189.

2.6.4 Alerting System

Alerting System is a Web-based application that focuses on finding problems in a dataset by checking for increases in the occurrence of particular keywords and combinations of highly-correlated keywords and categories. The Alerting System can be used to specify the categories and parameters to be used in analysis. The main functions of the Alerting System are to:

- ▶ Increase detection function:
Among a large number of keywords (including phrases), this function detects keywords that have recently been increasingly used. For example, among the Noun category keywords contained in the call center data, this function detects the keywords that are increasingly used through time for each week. The results of this analysis is useful for identifying future problems.

► Correlation detection function:

This function extracts combinations of keywords and subcategories that are highly correlated to each other. For example, by extracting the correlation between product category keywords and part/phrase category keywords from a manufacturer's call center data, expressions to describe product defects that are frequently used in association with a particular product can be extracted.

For more information on the Alerting System, read Chapter 9, “Alerting System” on page 221.

2.6.5 Online manual

The online manual of OmniFind Analytics Edition is a Web-based application that describes how to use the OmniFind Analytics Edition Text Miner application. It explains the Text Miner interface and how to use it. Also, it provides different steps involved in analysis such as selecting your category, generating search conditions, and using different views to analyze the results. We recommend using the online manual in conjunction with this book.

2.7 Integration with OmniFind Enterprise Edition

In OmniFind Analytics Edition, the input data to be analyzed, including text, should be prepared in Comma Separated Value (CSV) data format. Most of the relational databases and spreadsheet applications support exporting data to CSV format. If the data to be extracted is not in a data mart or spreadsheet, then the extraction process to CSV might become tedious.

In the situations where most of the data is located in different content repositories across the enterprise, such as IBM Content Manager, Document Manager, e-mails, Web sites, and portals, OmniFind Analytics Edition can be integrated with OmniFind Enterprise Edition (OEE), which can handle the data extraction part from the different repositories and feed this extracted data into OmniFind Analytics Edition for further analysis. IBM OmniFind Enterprise Edition is an enterprise search product that supports wide range of data sources, which include file systems, content repositories, databases, collaboration systems, intranet, extranet, and public-facing corporate Web sites. Moreover, it is the first commercially available UIMA based platform for processing unstructured information.

Figure 2-8 shows a high level view of the integration between OmniFind Analytics Edition and OmniFind Enterprise Edition.

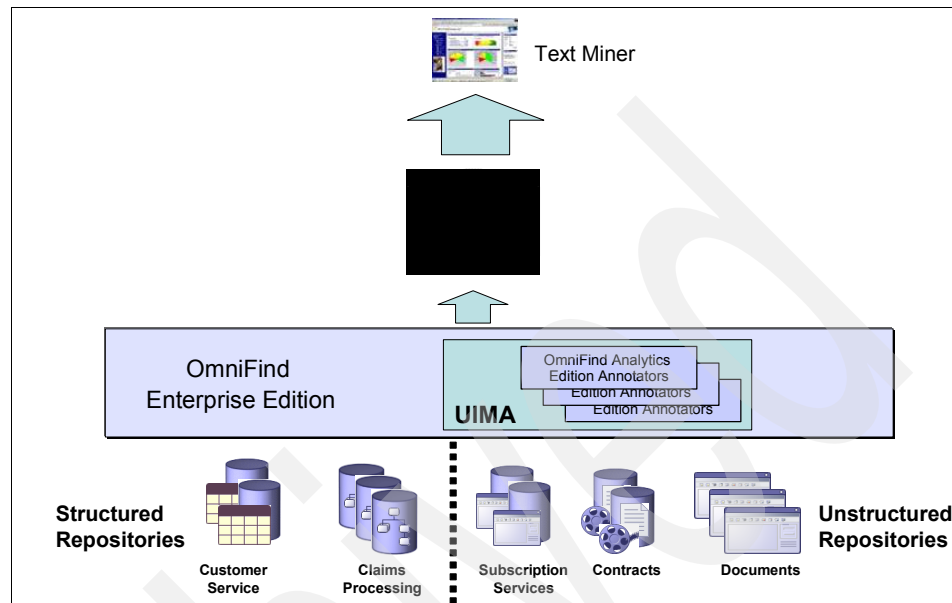


Figure 2-8 OmniFind Analytics Edition - OEE integration architecture

The integration is done by uploading the OmniFind Analytics Edition Natural Language Processing core in the form of UIMA annotators into OmniFind Enterprise Edition (after pear packaging it). These annotators can then process the data extracted by OmniFind Enterprise Edition and create output MIML files that are to be used by the OmniFind Analytics Edition Indexer and Text Miner. For more information on the integration, see 4.10, “Importing data using OmniFind Enterprise Edition” on page 86.

Installation and configuration

This chapter provides detailed steps required to successfully install and configure the OmniFind Analytics Edition (OAE) product.

We cover the following topics:

- ▶ Before you begin
- ▶ Installation and configuration for Windows platform
- ▶ Installation and configuration for AIX platform
- ▶ Verification of installation and configuration

3.1 Before you begin

Before you begin installing OmniFind Analytics Edition, make sure that all prerequisites are met for your environment. Always check for the latest system requirements before installation.

Note: The IBM OmniFind Analytics Edition installation will not work without a fully configured and accessible IBM WebSphere Application Server of at least version 6.0.2 with your operating environment.

Complete the following two steps before you begin:

1. Confirm that you have a working version of the IBM WebSphere Application Server installed.
2. Ensure that you have full administrator access rights to your IBM WebSphere Application Server environment.

OmniFind Analytics Edition can be installed in all IBM AIX® based operating system environments that are installed on a 64-bit architecture. Ensure that you have the necessary access rights to make changes to your root operating system environment.

3.1.1 Disk space and memory recommendation

We recommend at least 60 GB of free, available hard-disk space and 6 GB of internal memory that can be fully allocated for the preprocessing of data. The larger hard-disk space and memory requirements are needed when you begin to add source data to your IBM OmniFind Analytics Edition application. The more available disk space and memory you have, the better. This helps to reduce the amount of time that it takes when you begin to analyze your unstructured data.

If you plan to add additional disk space and memory at a later time, you can begin the installation portion with less system resources than what we suggest here.

3.1.2 System requirements

The following URL provides the latest update on system requirements:

<http://www.ibm.com/software/data/enterprise-search/omnifind-analytics/requirements.html>

Make sure that your environment meet these prerequisites.

The following operating systems are the minimum required for the OmniFind Analytics Edition installation:

- ▶ IBM AIX 5L™ v.5.3
- ▶ Microsoft® Windows® Server 2003 with Service Pack 1

The following software installations are the minimum required:

- ▶ IBM WebSphere Application Server Version 6.0.2
- ▶ Java Runtime Environment Version 1.4.2
- ▶ Microsoft Internet Explorer® Browser Version 6

Java Runtime Environment Version 1.4.2 is bundled with Installer.

The following hardware configurations are the minimum required:

- ▶ 32-bit CPU or 64-bit CPU for IBM AIX
- ▶ Intel® 32-bit CPU or AMD™ 32-bit CPU for Microsoft Windows Server® 2003
- ▶ Available internal memory of 6 GB recommended with 2 GB as the minimum
- ▶ Available hard-disk space of 30 GB recommended

It is always a good idea to check for the latest IBM documentation on releases and patch fix levels. Be sure to use the latest version levels for the platform you are installing. We recommend that you call your IBM Technical Support staff if you cannot find the required documentation for your platform version. The following IBM Web links assist in helping you find the appropriate information for worldwide support on IBM OmniFind products:

- ▶ <http://www.ibm.com/software/support>
- ▶ <http://www.ibm.com/software/data/enterprise-search/omnifind-analytics/support.html>
- ▶ <http://www.ibm.com/planetwide/index.html>

3.1.3 Installation file

If you do not have the OmniFind Analytics Edition installation file, obtain it from your IBM representative or download it from IBM support site. For the installation we perform for this book, we obtained the installation file named `TAKMI_INSTALL_WIN32-200708012019.zip`.

3.1.4 Installation process overview

If you have never performed an installation of IBM OmniFind Analytics Edition, follow the detailed steps in this chapter to complete a full installation. If you have previously completed an installation of IBM OmniFind Analytics Edition and are upgrading to the latest version of the product, you might want to contact IBM Software Support to verify that you have completed all the necessary prerequisite steps before continuing with the rest of the installation process.

The general installation process consists of the following main steps:

1. Install the software.
2. Configure the software.
3. Perform verification.

Depend on the operating system of your environment, you can follow the steps described in different sections of the chapter:

- ▶ For a Windows environment, follow the steps in 3.2, “Installation and configuration for Windows platform” on page 32.
- ▶ For an AIX environment, follow the steps in 3.3, “Installation and configuration for AIX platform” on page 42.

When everything is installed and configured, follow the steps in 3.4, “Verification of installation and configuration” on page 50 to ensure that everything is installed and configured correctly.

3.2 Installation and configuration for Windows platform

By now, you should have confirmed that the IBM WebSphere Application Server version 6.0.2 or later has been installed and configured correctly. You should also have confirmed that you have enough disk space and available physical memory to install and run the completed application. If you have not completed installing the IBM WebSphere Application Server or you do not have full administrator rights to install new applications, complete these prerequisites first.

3.2.1 Installing OmniFind Analytics Edition on Windows

To start the installation of OmniFind Analytics Edition, the WebSphere Application Server does not have to be running. We recommend completely stopping the WebSphere Application Server while performing the IBM OmniFind Analytics Edition installation.

To install OmniFind Analytics Edition, follow these steps:

1. Unzip the OmniFind Analytics Edition installation file to a temporary directory.

For our installation, the temporary directory we use is: c:\temp. The installation file we use is TAKMI_INSTALL_WIN32-200708012019.zip.

2. Start the installation process by running the executable file:

takmi setupwin32.exe

3. When the Welcome window displays (see Figure 3-1), click **Next**.

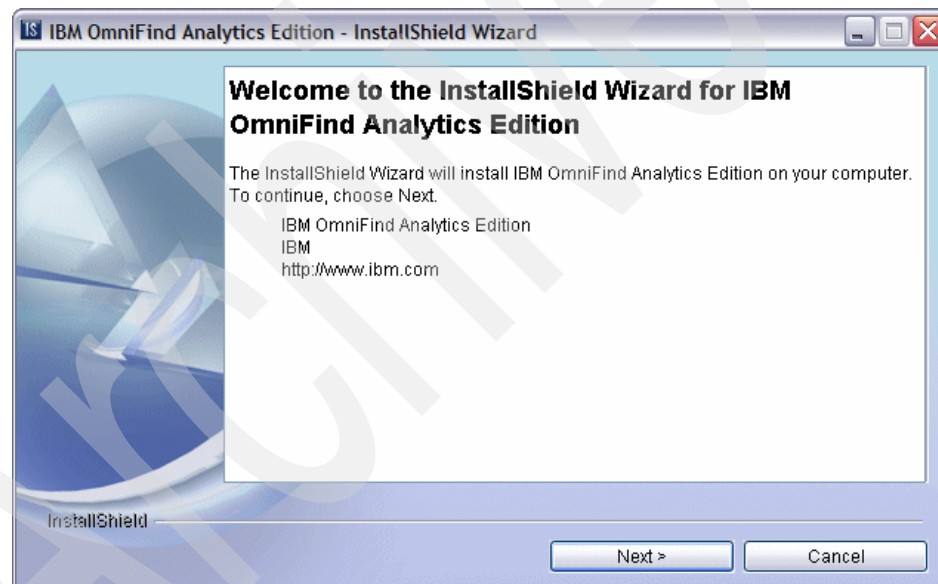


Figure 3-1 OmniFind Analytics Edition for Windows - Welcome window

4. When the License Agreement window displays (see Figure 3-2), select **I accept the terms and license agreement** and click **Next**.

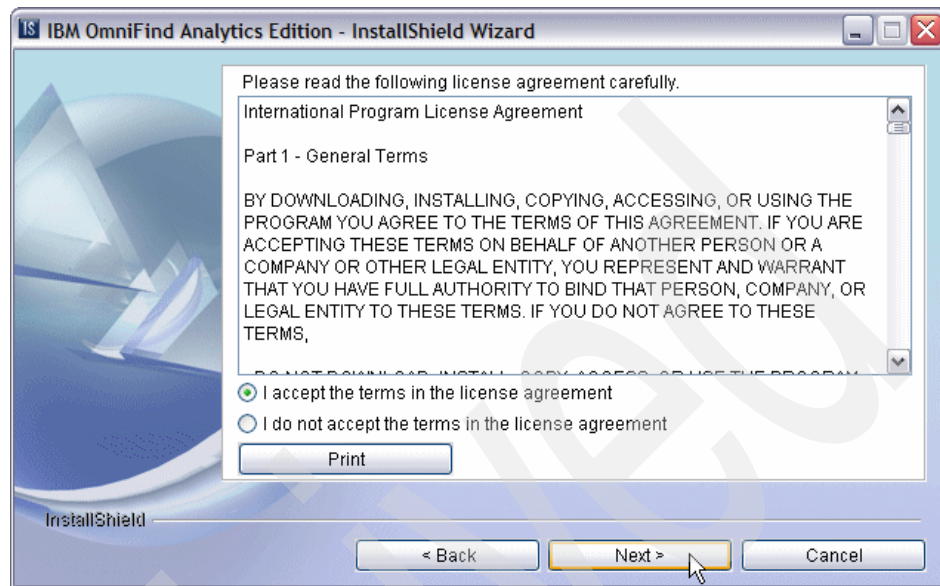


Figure 3-2 IBM OmniFind Analytics Edition for Microsoft Windows acceptance window

5. In the next window (see Figure 3-3), enter the installation path and click **Next**.
The default directory for a Microsoft Windows installation is: C:\Program Files\IBM\TAKMI. We recommend a shorter path. For our installation, we use C:\IBM\OmniFind.

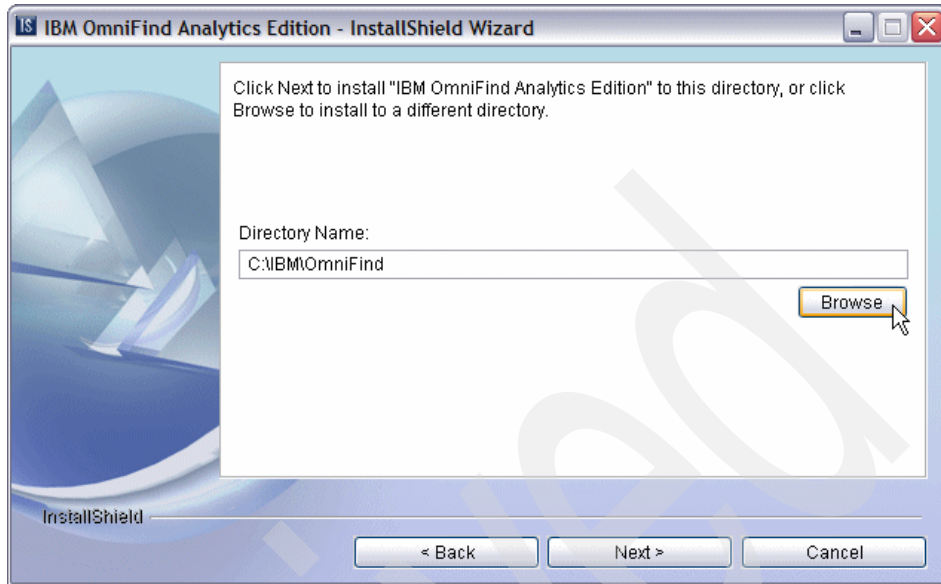


Figure 3-3 IBM OmniFind Analytics Edition for Microsoft Windows directory window

6. In the next window (see Figure 3-4), select **Typical** as the installation type and click **Next** to continue.

If you choose the Custom install, the system will prompt you to select the features that you want to install. The features include Program Files, Documents, and Sample Databases. The Program Files are the only files that you must choose for the install to work correctly. The other two features are optional and do not have to be installed. For simplification, we recommend that you select the Typical install.

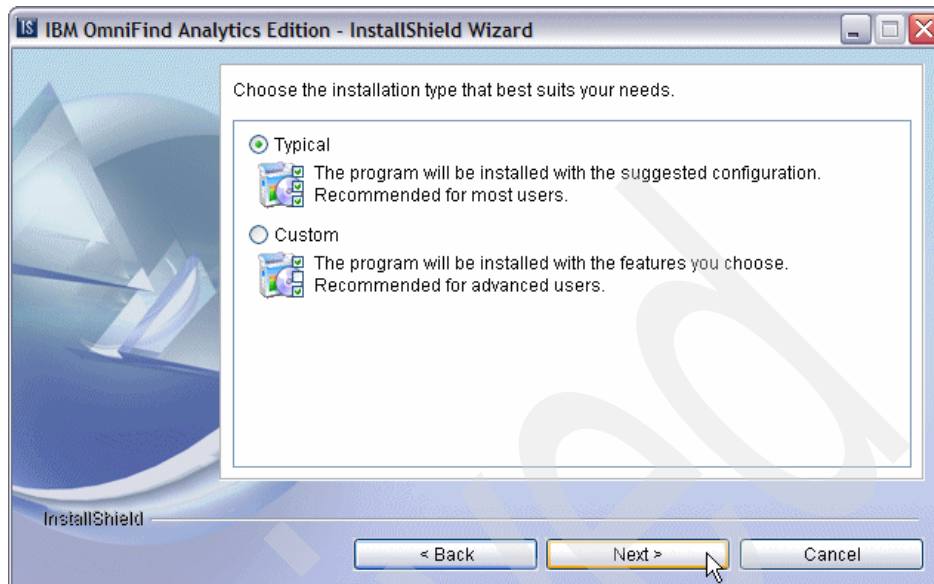


Figure 3-4 IBM OmniFind Analytics Edition for Microsoft Windows choice window

7. Review the Summary Information window (see Figure 3-5). Click **Install** to start the actual installation process.

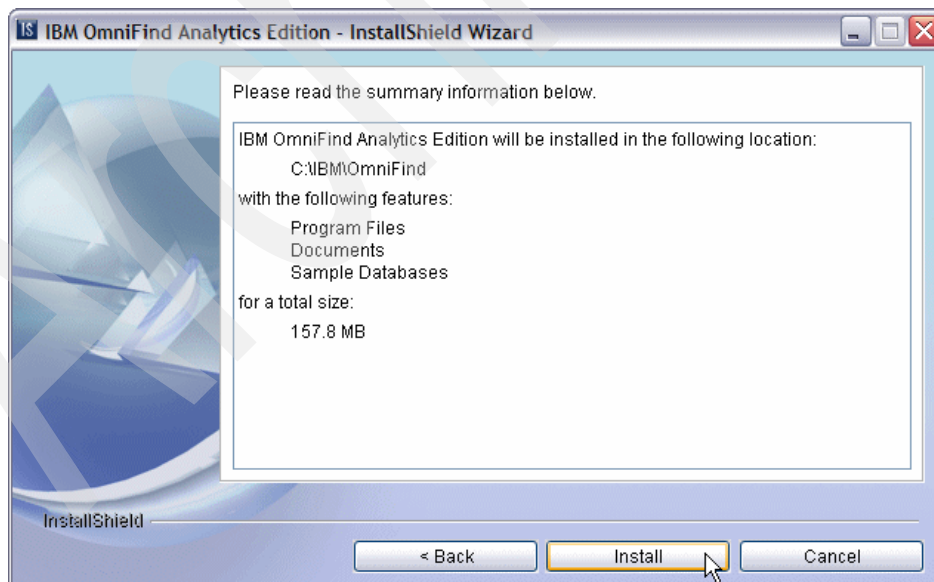


Figure 3-5 IBM OmniFind Analytics Edition for Microsoft Windows summary window

8. When you are done (see Figure 3-6), click **Finish**.

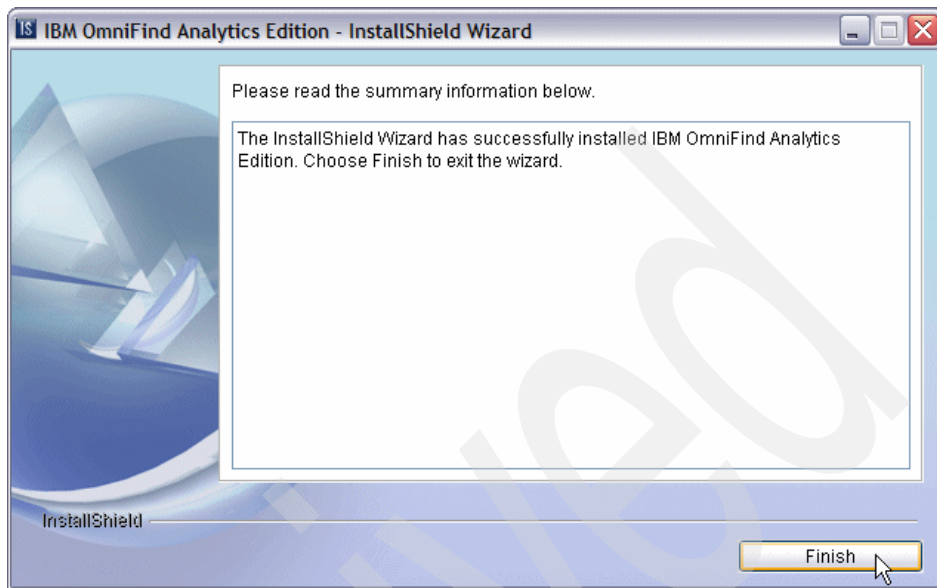


Figure 3-6 IBM OmniFind Analytics Edition for Microsoft Windows success window

When OmniFind Analytics Edition is installed, we can move on to configuration.

If you experience any installation problems, completely remove the directory where you installed OmniFind Analytics Edition and restart the installation process from the beginning.

3.2.2 Configuring OmniFind Analytics Edition on Windows

After you have completed the necessary installation steps, you must configure the OmniFind Analytics Edition environment to finalize the setup.

Follow these steps for OmniFind Analytics Edition configuration:

1. Start IBM WebSphere Application Server:

Select **Start** → **Control Panel** → **Administrative tools** → **Services**.

Right-click the IBM WebSphere Application Server entry and select **Start** (see Figure 3-7).

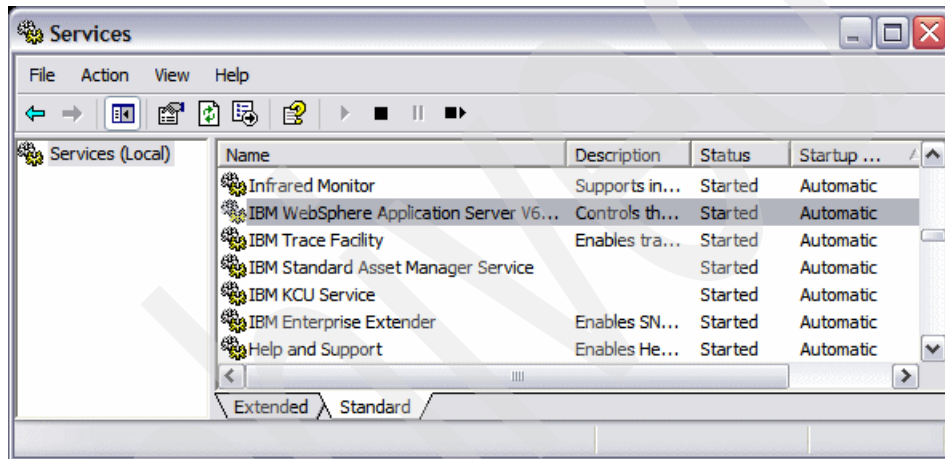
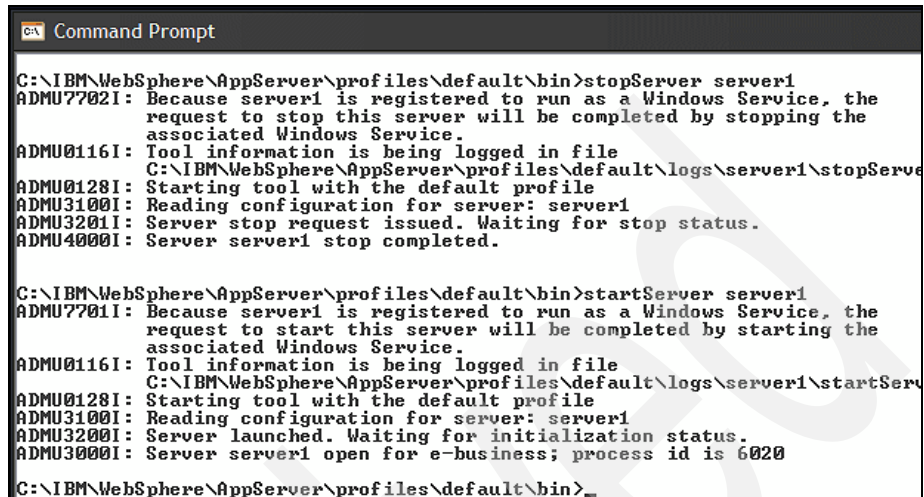


Figure 3-7 IBM WebSphere Application Server start

If for some reason you did not install the IBM WebSphere Application Server as a Windows service, then start the WebSphere Application Server from a command prompt:

- a. Go to the application server's bin directory. The default is usually:
C:\IBM\WebSphere\AppServer\profiles\default\bin.
- b. Start the server: **startServer server1**

Figure 3-8 shows the stopping and starting of WebSphere Application Server from a command prompt.



```
C:\IBM\WebSphere\AppServer\profiles\default\bin>stopServer server1
ADMU77021: Because server1 is registered to run as a Windows Service, the
request to stop this server will be completed by stopping the
associated Windows Service.
ADMU01161: Tool information is being logged in file
C:\IBM\WebSphere\AppServer\profiles\default\logs\server1\stopServe
ADMU01281: Starting tool with the default profile
ADMU31001: Reading configuration for server: server1
ADMU32001: Server stop request issued. Waiting for stop status.
ADMU40001: Server server1 stop completed.

C:\IBM\WebSphere\AppServer\profiles\default\bin>startServer server1
ADMU77011: Because server1 is registered to run as a Windows Service, the
request to start this server will be completed by starting the
associated Windows Service.
ADMU01161: Tool information is being logged in file
C:\IBM\WebSphere\AppServer\profiles\default\logs\server1\startServe
ADMU01281: Starting tool with the default profile
ADMU31001: Reading configuration for server: server1
ADMU32001: Server launched. Waiting for initialization status.
ADMU30001: Server server1 open for e-business; process id is 6020

C:\IBM\WebSphere\AppServer\profiles\default\bin>
```

Figure 3-8 IBM WebSphere Application Server start

2. This step is an optional step. If you would like to double-check that your environment is set up properly, or if something has gone wrong with your first installation and you have to go through the installation again, perform this step. Otherwise, you can skip this step.

Locate and open the TAKMI.jacl file in a text editor and verify that your environment variables, TAKMI_HOME, UIMA_HOME, and CLASSPATH are all set up correctly. Make sure that the CLASSPATH has been updated with the two entries.

The TAKMI.jacl file is located in the <TAKMI_HOME>\ear directory, where TAKMI_HOME is the installation directory for your OmniFind Analytics Edition file. For our installation, the file is located at: C:\IBM\OmniFind\ear directory.

Figure 3-9 shows a portion of the TAKMI.jacl file and the correct environment variables that should have been set during the installation.

To check your environment variable settings:

Select **Start** → **Control Panel** → **System** → **System Properties** → **Advanced Tab** → **Environment Variables**.

You should also confirm that the TAKMI.jacl file has the correct .ear files listed in the \$AdminApp install directory on the bottom of this file.

```

# Java Virtual Machine
#

set jvm [$AdminConfig list JavaVirtualMachine [$AdminConfig getid
/Server:server1/]]
set sps [lindex [$AdminConfig showAttribute $jvm systemProperties] 0]

foreach sp $sps {
    if {[lindex [$AdminConfig show $sp] 0] == "name TAKMI_HOME" ||
        [lindex [$AdminConfig show $sp] 0] == "name uima.home" ||
        [lindex [$AdminConfig show $sp] 0] == "name ws.ext.dirs" } {
        $AdminConfig remove $sp
    }
}

$AdminConfig modify $jvm {{systemProperties {{name TAKMI_HOME} {value
"C:\IBM\OmniFind"}}}}}
$AdminConfig modify $jvm {{systemProperties {{name uima.home} {value
"C:\IBM\OmniFind\uima"}}}}}
$AdminConfig modify $jvm {{systemProperties {{name ws.ext.dirs} {value
"C:\IBM\OmniFind\lib;C:\IBM\OmniFind\uima\lib;C:\IBM\OmniFind\uima\component
s\TAKMI_NLP\lib;C:\IBM\OmniFind\uima\components\jsa\lib" }}}}}

...

$AdminApp install ./TAKMI_ALERT.ear {-server server1}
$AdminApp install ./TAKMI_DIC.ear {-server server1}
$AdminApp install ./TAKMI_MANUAL.ear {-server server1}
$AdminApp install ./TAKMI_MINER.ear {-server server1}

...

```

Figure 3-9 Confirm TAKMI_HOME, UIMA_HOME and classpath are set correctly

For our installation, Figure 3-10 shows the correct settings for the environment variables TAKMI_HOME and UIMA_HOME.

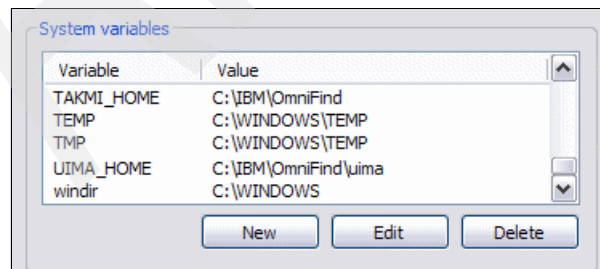


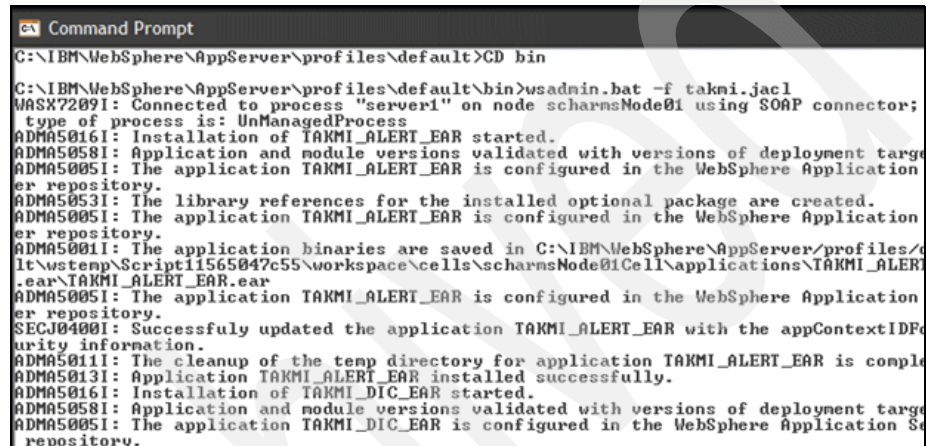
Figure 3-10 TAKMI_HOME and UIMA_HOME environment variables

3. Execute the following command to install the four TAKMI applications (TAKMI_ALERT, TAKMI_DIC, TAKMI_MANUAL, and TAKMI_MINER) from the TAKMI*.ear files:

```
<WAS_HOME>\bin\wsadmin.bat -f .\TAKMI.jacl
```

<WAS_HOME> is where IBM WebSphere Application Server directory is installed. For our installation, <WAS_HOME> is located at C:\IBM\WebSphere\AppServer\profiles\default directory.

Figure 3-11 shows the execution of the command.



```
CA Command Prompt
C:\IBM\WebSphere\AppServer\profiles\default>CD bin
C:\IBM\WebSphere\AppServer\profiles\default\bin>wsadmin.bat -f takmi.jacl
WASX72091: Connected to process "server1" on node scharmsNode01 using SOAP connector;
type of process is: UnManagedProcess
ADMA50161: Installation of TAKMI_ALERT_EAR started.
ADMA50581: Application and module versions validated with versions of deployment target
ADMA50051: The application TAKMI_ALERT_EAR is configured in the WebSphere Application
er repository.
ADMA50531: The library references for the installed optional package are created.
ADMA50051: The application TAKMI_ALERT_EAR is configured in the WebSphere Application
er repository.
ADMA50011: The application binaries are saved in C:\IBM\WebSphere\AppServer\profiles\de
lt\ustemp\Script11565047c55\workspace\cells\scharmsNode01Cell\applications\TAKMI_ALERT
.ear\TAKMI_ALERT_EAR.ear
ADMA50051: The application TAKMI_ALERT_EAR is configured in the WebSphere Application
er repository.
SECJ04001: Successfully updated the application TAKMI_ALERT_EAR with the appContextIDFo
urity information.
ADMA50111: The cleanup of the temp directory for application TAKMI_ALERT_EAR is comple
ADMA50131: Application TAKMI_ALERT_EAR installed successfully.
ADMA50161: Installation of TAKMI_DIC_EAR started.
ADMA50581: Application and module versions validated with versions of deployment target
ADMA50051: The application TAKMI_DIC_EAR is configured in the WebSphere Application Se
repository.
```

Figure 3-11 TAKMI.jacl script that installs .ear files

4. Stop and restart WebSphere Application Server.
5. Verify that the TAKMI applications are installed on WebSphere Application Server:
 - a. Launch the WebSphere Application Server's administrative console by selecting **Start** → **Programs** → **IBM WebSphere** → **Application Server V6** → **Profiles** → **default** → **Administrative console**.
 - b. Expand **Applications** → **Enterprise Applications**.
 - c. Verify that the applications are installed. See Figure 3-12.

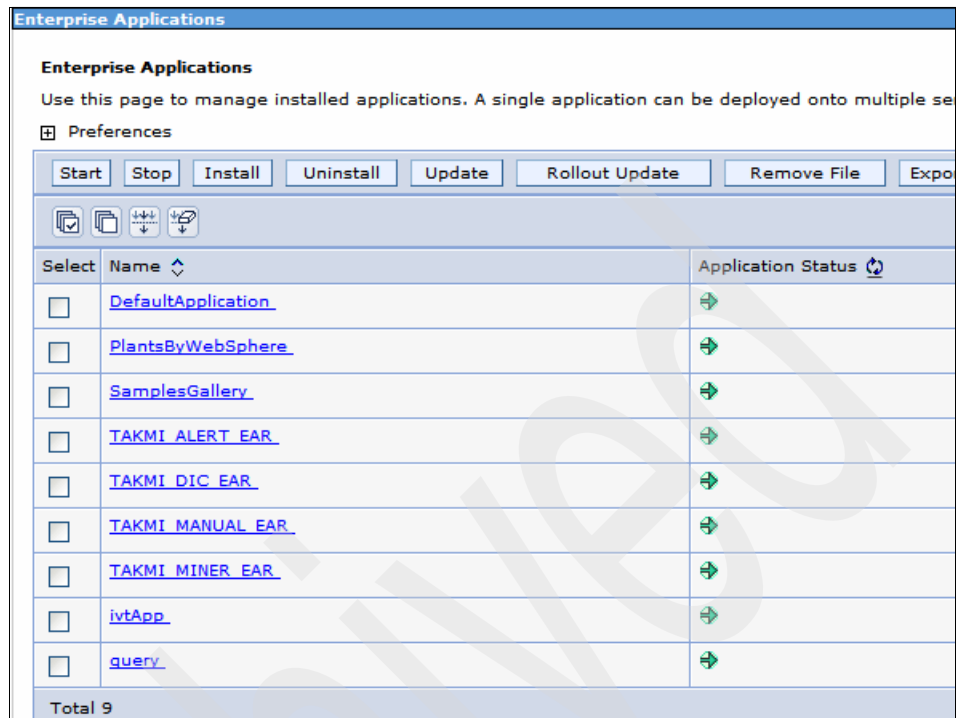


Figure 3-12 New TAKMI Enterprise Applications

We have now completed all steps required for the installation and configuration of OmniFind Analytics Edition. Proceed to 3.4, “Verification of installation and configuration” on page 50 to further verify that all applications are running successfully.

3.3 Installation and configuration for AIX platform

By now, you should have confirmed that the IBM WebSphere Application Server is at least at Version 6.02 level and it has been installed and configured correctly. You should also have confirmed that you have enough disk space and available physical memory to install and run the completed application. If you have not completed installing the IBM WebSphere Application Server or you do not have full administrator rights to install new applications, complete these prerequisites first.

To check your internal memory:

1. Log in as ROOT or log in an account with ROOT privileges.
2. Execute the following command to see the internal memory:

```
bootinfo -r
```

3.3.1 Installing OmniFind Analytics Edition on AIX

To start the installation of OmniFind Analytics Edition, the WebSphere Application Server does not have to be running. We recommend completely stopping the WebSphere Application Server while performing the IBM OmniFind Analytics Edition installation.

To install OmniFind Analytics Edition, follow these steps:

1. Log in as ROOT or log in an account with the ROOT privileges. and untar or unzip the OmniFind Analytics Edition installation file.
2. Execute the following command to start the installation process:

```
./takmisetupaix.bin
```

3. The Welcome window displays. See Figure 3-13. Click **Next**.

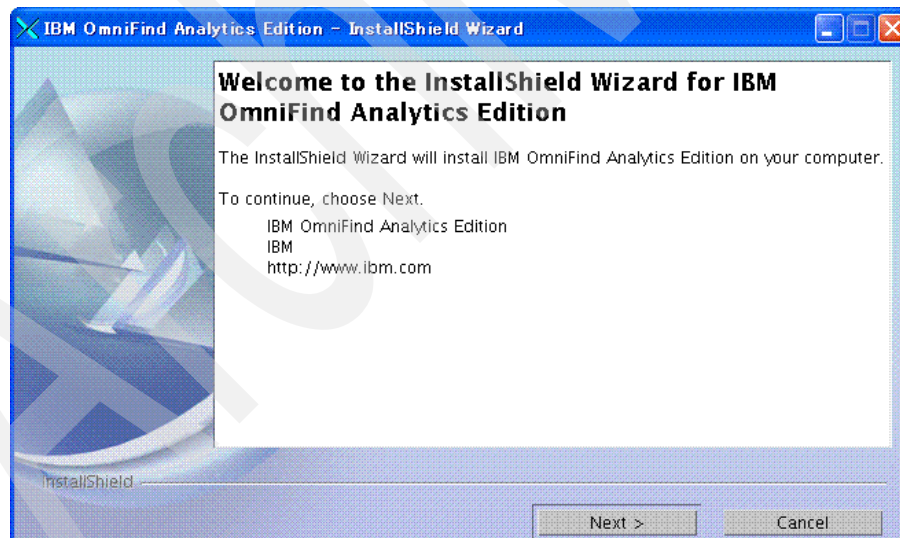


Figure 3-13 IBM AIX Welcome window for installation

4. When the License Agreement window displays (see Figure 3-14), select **I accept the terms in the license agreements** and click **Next**.

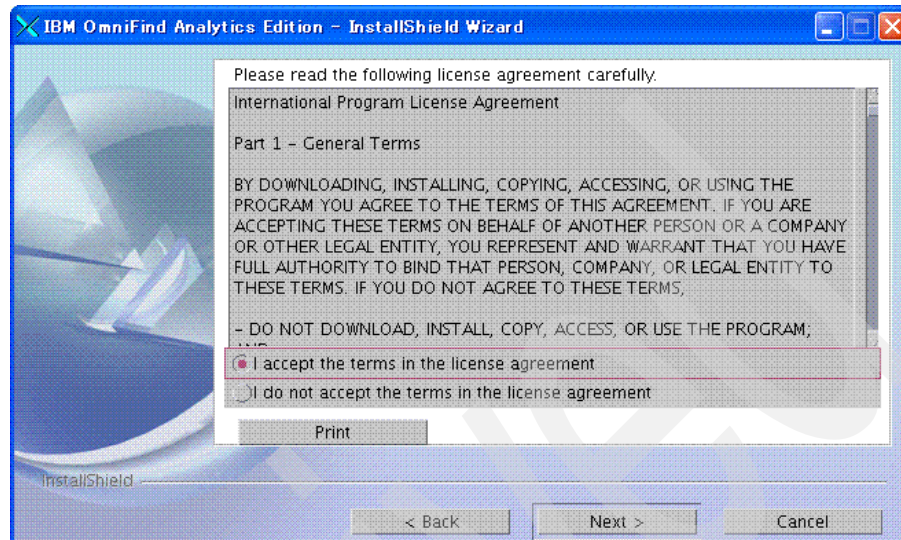


Figure 3-14 IBM AIX license acceptance window

5. Enter the installation directory (see Figure 3-15) and click **Next**.
For our installation, we use /opt/IBM/OAE as the installation directory.

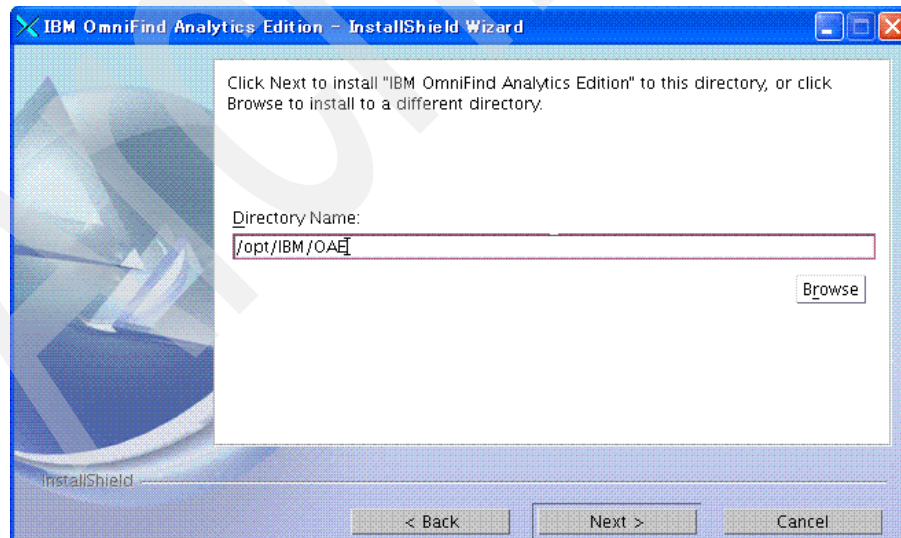


Figure 3-15 IBM AIX installation path choice

6. Choose **Typical** as the installation type (see Figure 3-16) and click **Next**.

If you choose the Custom install, the system will prompt you to select the features that you want to install. The features include Program Files, Documents, and Sample Databases. The Program Files are the only files that you must choose for the install to work correctly. The other two features are optional and do not have to be installed. For simplification, we recommend that you select the Typical install.

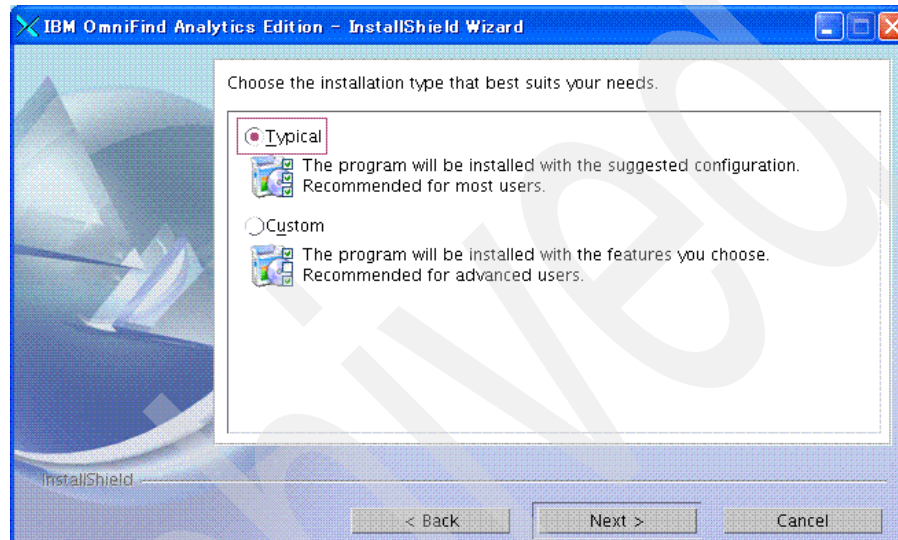


Figure 3-16 IBM AIX typical or custom installation

7. Review the Summary Information window (see Figure 3-17). Click **Install** to start the actual installation process.

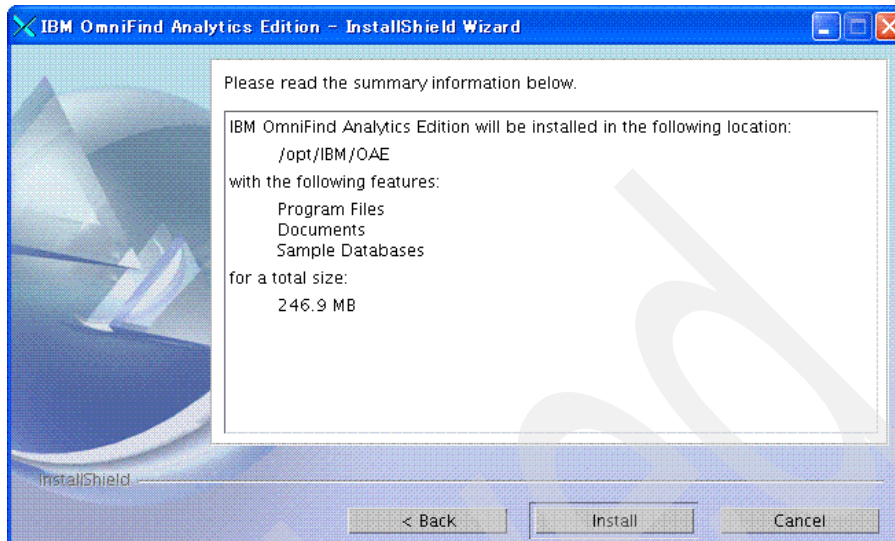


Figure 3-17 IBM AIX install confirmation window: click *Install* to complete

8. When the installation is completed, click **Finish** (see Figure 3-18).

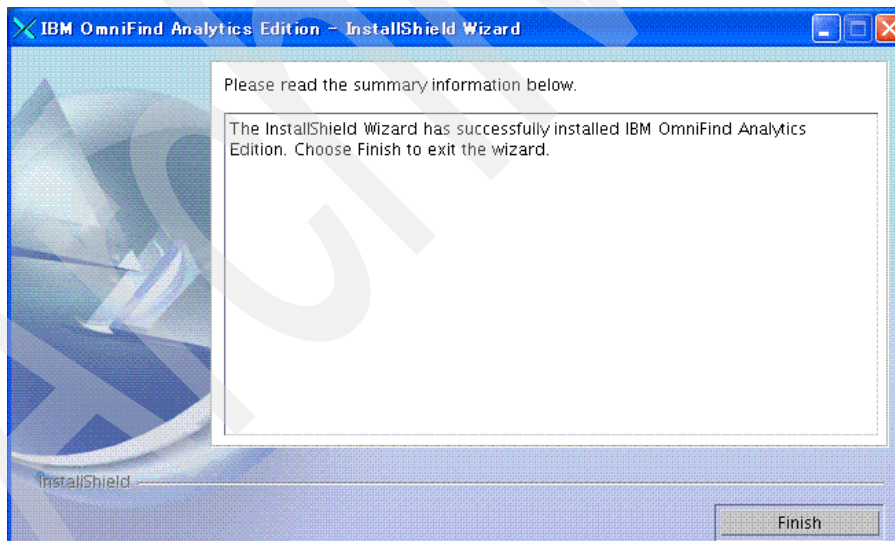


Figure 3-18 IBM AIX finished installing prompt

9. The installation program now asks you to log out of the AIX environment to allow the installation files to set correctly (see Figure 3-19). Click **Finish** and then log out and log back into your AIX environment.

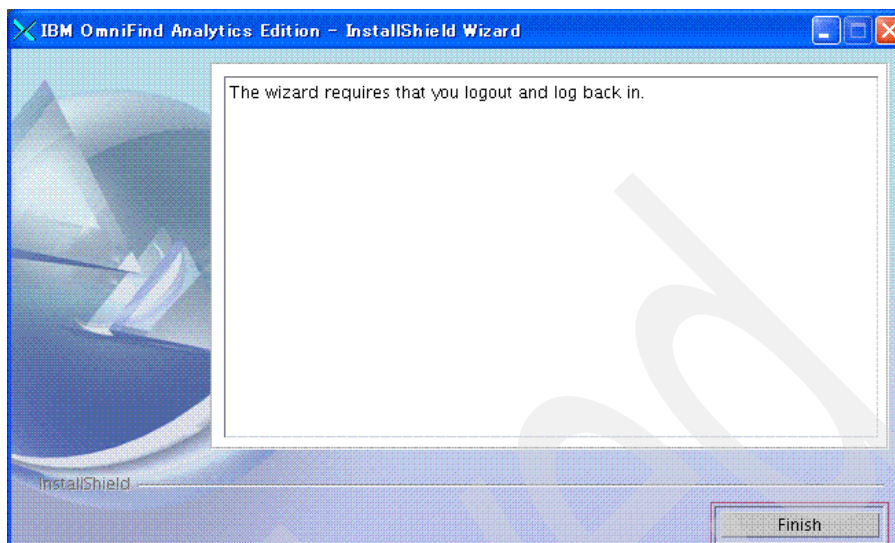


Figure 3-19 IBM AIX finish window

10. This step is an optional step. If you would like to double-check that your environment is set up properly, or if something has gone wrong with your first installation and you have to go through the installation again, perform this step. Otherwise, you can skip this step.

After logging back in, verify that the environment variables have been set properly. For our installation, our environment is shown in Figure 3-20.

Make sure that your `/etc/profile` correctly reflects the correct environment variables. For our installation, the variables have the following settings:

```
TAKMI_HOME = /opt/IBM/OAE
UIMA_HOME = /opt/IBM/OAE/uima
```

You might have to modify your `/etc/profile` to ensure that the install path of `/opt/IBM/OAE` is identified in `PATH` and `LIBPATH`.

To view your environment variables, use the following command:

```
env | more
```

To set the environment variables, use the command:

```
set | grep TAKMI_HOME
```

Confirm that the permissions for all new application files in the `TAKMI_HOME` directory have been set correctly to have `ROOT` access.

```
# bos530 src/bos/etc/profile/profile 1.20

# Begin INSTALLSHIELD Environment Variable Section
# Do not edit this section manually.
# var 0 TAKMI_HOME=/opt/IBM/OAE
TAKMI_HOME=/opt/IBM/OAE
export TAKMI_HOME

# var 1 :
PATH=`echo $PATH`:/opt/IBM/OAE/bin:/opt/IBM/OAE/uima/components/jsa/lib
if [ -z "`echo $PATH`" ]
then
PATH=/opt/IBM/OAE/bin:/opt/IBM/OAE/uima/components/jsa/lib
else
PATH=`echo $PATH`:/opt/IBM/OAE/bin:/opt/IBM/OAE/uima/components/jsa/lib
fi
export PATH

# var 1 :
LIBPATH=`echo $LIBPATH`:/opt/IBM/OAE/uima/components/jsa/lib
if [ -z "`echo $LIBPATH`" ]
then
LIBPATH=/opt/IBM/OAE/uima/components/jsa/lib
else
LIBPATH=`echo $LIBPATH`:/opt/IBM/OAE/uima/components/jsa/lib
fi
export LIBPATH

# var 0 UIMA_HOME=/opt/IBM/OAE/uima
UIMA_HOME=/opt/IBM/OAE/uima
export UIMA_HOME

# End INSTALLSHIELD Environment Variable Section
```

Figure 3-20 IBM AIX environment variables to confirm

11. After the environment variables settings are verified, start the WebSphere Application Server.

From the WebSphere Application Server's bin directory, run:

```
./startServer.sh server1
```

For our installation, the WebSphere Application Server's bin directory is located at: /usr/IBM/WebSphere/AppServer/bin.

To check the server status, run:

```
./serverStatus.sh -all
```

3.3.2 Configuring OmniFind Analytics Edition on AIX

After you have completed the necessary installation steps, you must configure the OmniFind Analytics Edition environment to finalize the setup.

Follow these steps for OmniFind Analytics Edition configuration:

1. Locate and open the TAKMI.jacl file to verify that the setting is correct.

Figure 3-21 shows a portion of the TAKMI.jacl file from our installation.


```

# TAKMI.jacl
$AdminConfig modify $jvm {{systemProperties {{name uima.home} {value "/opt/IBM/OAE/uima"}}}}}
$AdminConfig modify $jvm {{systemProperties {{name ws.ext.dirs} {value
"/opt/IBM/OAE/lib:/opt/IBM/OAE/uima/lib:/opt/IBM/OAE/uima/components/TAKMI_NLP/lib:/o
pt/IBM/OAE/uima/components/jsa/lib"}}}}}

#
# Enterprise Application

set appManager [$AdminControl queryNames type=ApplicationManager,process=server1,*]
set apps [$AdminApp list]

foreach app $apps {
    if {$app == "TAKMI_ALERT_EAR" ||
        $app == "TAKMI_DIC_EAR" ||
        $app == "TAKMI_MANUAL_EAR" ||
        $app == "TAKMI_MINER_EAR"} {
        if {[AdminControl completeObjectName type=Application,name=$app,process=server1,*]
            != ""} {
            $AdminControl invoke $appManager stopApplication $app
        }
        $AdminApp uninstall $app
    }
}

$AdminApp install ./TAKMI_ALERT.ear {-server server1}
$AdminApp install ./TAKMI_DIC.ear {-server server1}
$AdminApp install ./TAKMI_MANUAL.ear {-server server1}
$AdminApp install ./TAKMI_MINER.ear {-server server1}

$AdminConfig save

```

Figure 3-21 IBM AIX TAKMI.jacl script

OmniFind Analytics Edition is packaged with four J2EE compatible applications that make up the user interface. Confirm that you have the following .jacl file and .ear files in your <TAKMI_HOME>\ear directory:

```

TAKMI.jacl
TAKMI_MINER.ear
TAKMI_DIC.ear
TAKMI_ALERT.ear
TAKMI_MANUAL.ear

```

<TAKMI_HOME> is where you installed OmniFind Analytics Edition. For our installation, <TAKMI_HOME> is the /opt/IBM/OAE directory.

2. Run the following command to deploy the .ear files to WebSphere Application Server:

```
<WAS_HOME>/bin/wsadmin.sh -f TAKMI.jacl
```

<WAS_HOME> is the installation root directory of WebSphere Application Server. For our installation, we use:

```
/usr/IBM/WebSphere/AppServer/bin/wsadmin.sh -f TAKMI.jacl
```

3. Open the WebSphere Application Server Administrative Console to ensure that the applications are properly deployed.

Also check to make sure that the Java Virtual Machine has been mapped to the current TAKMI_HOME installation path.

3.4 Verification of installation and configuration

Now is time to verify that you can access all four application that you have just deployed to the WebSphere Application Server. The normal installation for the WebSphere Application Server would have been typically installed using the default port number settings of 9080. So unless you choose to install using a different port number or you have other applications that are using this port number, you will use 9080 as the port number.

To ensure that the installation was completed successfully, verify that the following applications can be launched:

- ▶ Text Miner (TAKMI_MINER)
- ▶ Dictionary Editor (TAKMI_DIC)
- ▶ Alerting System (TAKMI_ALERT)
- ▶ Online Manual (TAKMI_MANUAL)

Text Miner (TAKMI_MINER) application verification

From a Web browser, enter the following URL to verify that Text Miner (TAKMI_MINER) is running properly:

`http://<Server Name:Port Number>/TAKMI_MINER`

For our installation, we use `http://localhost:9080/TAKMI_MINER` to verify. → Figure 3-22 shows the user interface of the application.

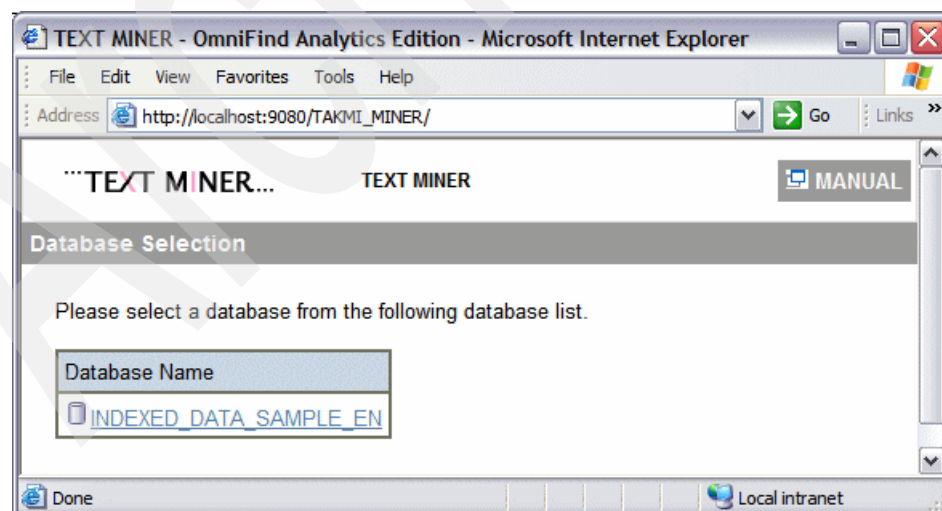


Figure 3-22 TAKMI_MINER application

During the installation process of IBM OmniFind Analytics Edition, a sample database is installed. It is located in the default application home directory. For our installation, it is in the following directory:

C:\IBM\OmniFind\databases\INDEXED_DATA_SAMPLE_EN

Dictionary Editor (TAKMI_DIC) application verification

From a Web browser, enter the following URL to verify that Dictionary Editor (TAKMI_DIC) is running properly:

http://<Server Name:Port Number>/TAKMI_DIC

For our installation, we use http://localhost:9080/TAKMI_DIC to verify. Figure 3-23 shows the user interface of the application.

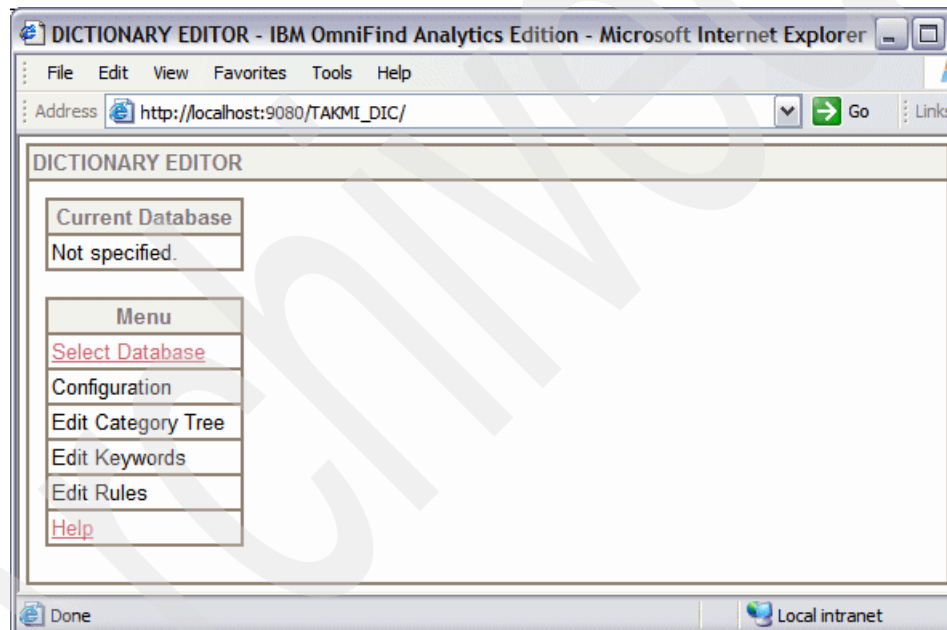


Figure 3-23 TAKMI_DIC application

Alerting System (TAKMI_ALERT) application verification

From a Web browser, enter the following URL to verify that the Alerting System (TAKMI_ALERT) is running properly:

`http://<Server Name:Port Number>/TAKMI_ALERT`

For our installation, we use `http://localhost:9080/TAKMI_ALERT` to verify. Figure 3-24 shows the user interface of the application.

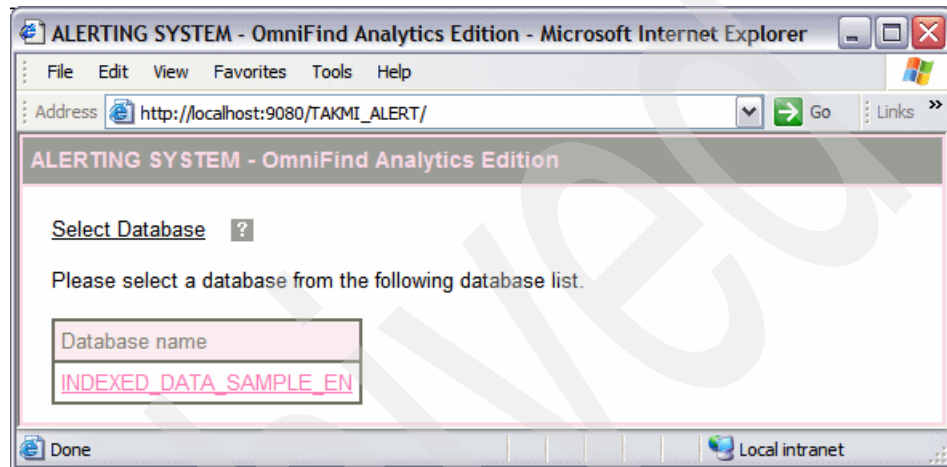


Figure 3-24 TAKMI_ALERT application

Online Manual (TAKMI_MANUAL) application verification

From a Web browser, enter the following URL to verify that the online Manual (TAKMI_MANUAL) is running properly:

`http://<Server Name:Port Number>/TAKMI_MANUAL/en/miner`

For our installation, we use `http://localhost:9080/TAKMI_MANUAL/en/miner` to verify. Figure 3-25 shows the user interface of the application.

You can also reach the online manual from the Text Miner application by launching the application and clicking the **MANUAL** button on the upper right corner of the page.

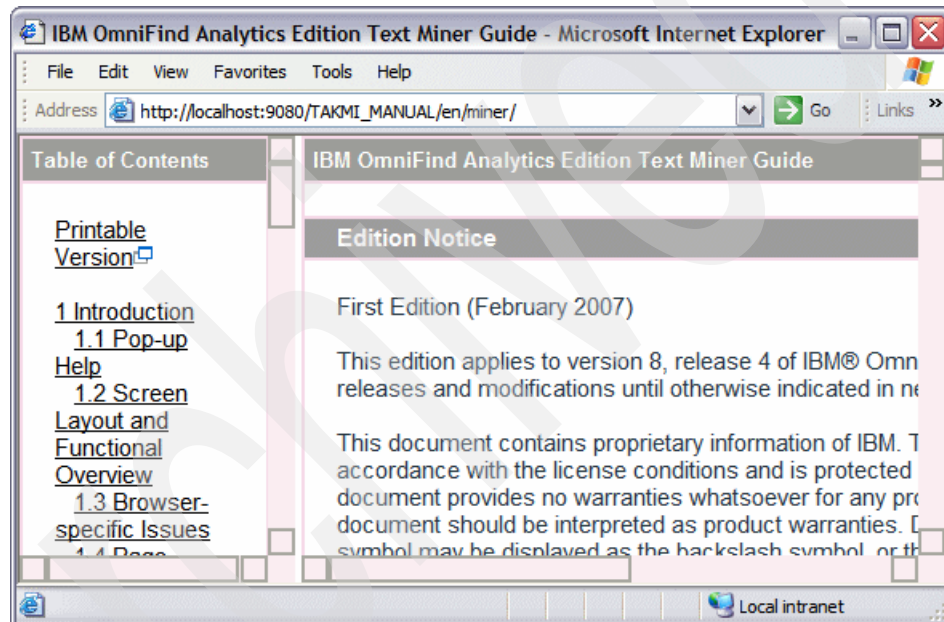


Figure 3-25 TAKMI_MANUAL application

Data ingestion and indexing

In this chapter, we describe a step-by-step approach to import and index data by OmniFind Analytics Edition (OAE). We introduce the steps required to build an analytics index by using a sample database from National Highway Traffic Safety Administration (NHTSA).

We cover the following topics:

- ▶ OmniFind Analytics Edition preprocessing overview
- ▶ Database directory creation and registration
- ▶ Preparation of CSV files
- ▶ Database category creation
- ▶ Generating ATML files
- ▶ Natural language processing
- ▶ Indexing
- ▶ Accessing Text Miner application
- ▶ Deleting data
- ▶ Importing data using OmniFind Enterprise Edition
- ▶ Quick reference for data ingestion and indexing

4.1 OmniFind Analytics Edition preprocessing overview

Before the source data can be applied for analysis, OmniFind Analytics Edition has to apply natural language preprocessing to discover hidden information through batch processing.

Figure 4-1 shows the flow of data processing and highlights the application of various batch processes required to generate indexes used by OmniFind Analytics Edition.

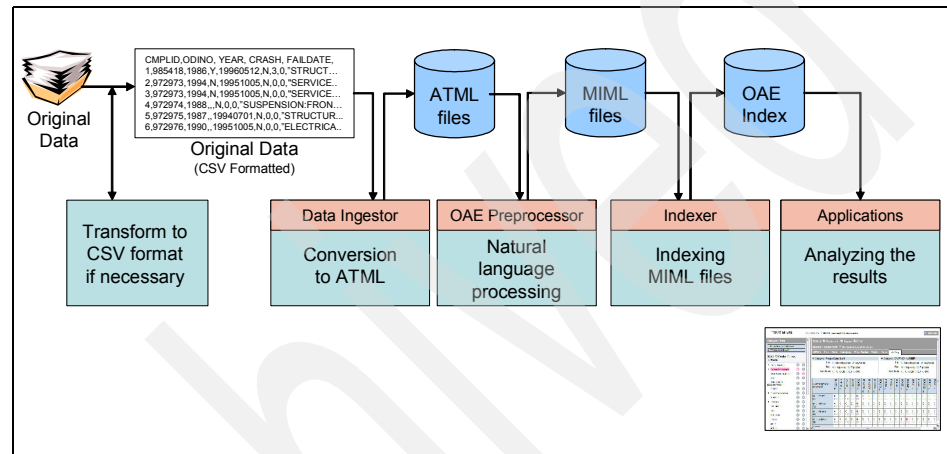


Figure 4-1 Overview of the data flow processing in OmniFind Analytics Edition

Figure 4-2 outlines the preprocessing steps that are applied to the source data.

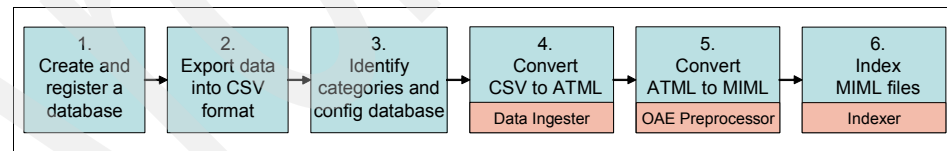


Figure 4-2 Sequential steps for preprocessing of source data

Next we explain the preprocessing steps, as follows:

1. Database directory creation and registration:

Create the database directory. Manually edit the global configuration file, `global_config.xml`, with the database name that represents the indexed data. Refer to 4.2, "Database directory creation and registration" on page 58 for more details.

2. Export data into CSV format:

As a prerequisite to use the OmniFind Analytics Edition's batch processes, the source data has to be in CSV formatted files. You can use Microsoft Excel® application or database export commands to generate CSV files for the source data. See 4.3, "Preparation of CSV files" on page 60 for more details.

3. Identify categories and configuration database:

Understand the schema of the source data and register category names in `category_tree.xml`. Also mark the identified category names in the `database_config.xml` file as `reserved_by_system`. See 4.4, "Database category creation" on page 63 for more details.

4. Convert CSV to ATML using Data Ingestor:

Invoke the OmniFind Analytics Edition batch process to convert the CSV formatted files into OmniFind Analytics Edition's proprietary file format called Advanced Text analytic Markup Language (ATML). See 4.5, "Generating ATML files" on page 70 for more details.

5. Perform natural language processing using OmniFind Analytics Edition Preprocessor:

Apply OmniFind Analytics Edition's natural language annotators to convert ATML files into an OmniFind Analytics Edition's proprietary file format called Mining Markup Language (MIML). The MIML files contains the source data along with the extra metadata that are discovered during the natural language processing. See 4.6, "Natural language processing" on page 78 for details.

6. Index MIML files using Indexer:

Using Indexer to generate OmniFind Analytics Edition's analytical indexes from the MIML files. See 4.7, "Indexing" on page 81 for more details.

7. Analyze the results using Text Miner:

This is technically not a preprocessing step. However, we include it here for a complete picture. After the index is generated from the previous step, the navigational features provided in the Text Miner application can be used to browse and discover the business insights that are scattered across the source data. In 4.8, "Accessing Text Miner application" on page 83, we provide instructions to access the Text Miner application. To learn features provided by the application, see Chapter 5, "Text Miner mining basics" on page 87 and Chapter 6, "Text Miner advanced" on page 141.

For the remaining sections of the chapter, we explain in detail the steps required to preprocess data. For a quick reference of all the associated commands and files to be updated, go directly to 4.11, "Quick reference for data ingestion and indexing" on page 86.

4.2 Database directory creation and registration

A *database* in OmniFind Analytics Edition is presented by a directory that holds physical files that represents the index, MIML files, dictionaries, rules, ATML files, and global configuration files. The structure of the directory tree is proprietary to OmniFind Analytics Edition's implementation.

Note: A database in OmniFind Analytics Edition context is not the same as a database in relational database context. Ideally, for each OmniFind Analytics Edition project or dataset, there is one database directory. The information in the database directory are stored as non-relational-database files.

Figure 4-3 shows a screen capture of the directory tree assigned for the sample database called NHTSA_SAMPLE. All the files under the directory are collectively termed as *database resource files*.

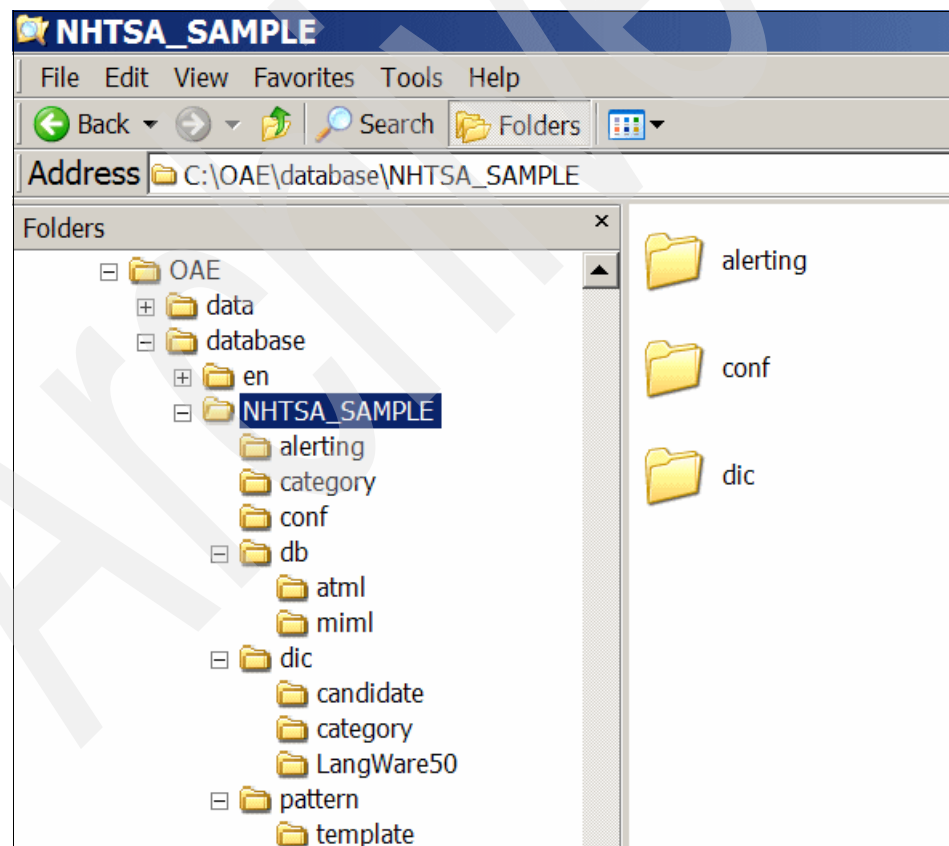


Figure 4-3 Directory structure of NHTSA_SAMPLE database

Note: <TAKMI_HOME> and UTF-8

- ▶ <TAKMI_HOME> is the directory where OmniFind Analytics Edition is installed. To determine the installed directory, review your system environment variables. For a default installation, <TAKMI_HOME> is set to C:\Program Files\IBM\takmi.
- ▶ All the XML files in OmniFind Analytics Edition should be saved in UTF-8 format. If you are using a Windows application such as Notepad, there is an option to save the files in UTF-8 format.

To create a new database, OmniFind Analytics Edition provides a sample directory to be used as a template. We offer the following instructions to guide you in creating the resource files for sample database called NHTSA_SAMPLE:

1. Copy the directory <TAKMI_HOME>\resource\database.template\en to <TAKMI_HOME>\databases.
2. Rename the directory <TAKMI_HOME>\databases\en to a name that will represent your source data.

In our scenario, we rename “en” to “NHTSA_SAMPLE” and the database directory path becomes <TAKMI_HOME>\databases\NHTSA_SAMPLE.
3. Register the created database in the global configuration file. Edit the global configuration file called <TAKMI_HOME>\conf\global_config.xml and add the lines as highlighted in Example 4-1.

Example 4-1 global_config.xml with database relative to the install directory

```
<?xml version="1.0" encoding="UTF-8"?>
<global_config>
<params>
<param name="language" value="en"/>
</params>
<database_entries>
<database_entry name="INDEXED_DATA_SAMPLE_EN" path_type="relative"
path="databases/INDEXED_DATA_SAMPLE_EN"/>
<database_entry name="NHTSA_SAMPLE" path_type="relative"
path="databases/NHTSA_SAMPLE"/>
</database_entries>
</global_config>
```

4. The processed OmniFind Analytics Edition indexes require large amounts of hard disk space. The OmniFind Analytics Edition indexes can be built on network drives or file systems and still be accessed by the Text Miner application running on a remote server. Refer to 2.4, “Topologies supported” on page 20 to learn about the various OmniFind Analytics Edition topologies. In the highlighted text shown in Example 4-2, we specify the absolute path of the remote database directory.

Example 4-2 global_config.xml with database absolute to the install directory

```
<?xml version="1.0" encoding="UTF-8"?>
<global_config>
<params>
<param name="language" value="en"/>
</params>
<database_entries>
<database_entry name="INDEXED_DATA_SAMPLE_EN" path_type="relative"
path="databases/INDEXED_DATA_SAMPLE_EN"/>
<database_entry name="NHTSA" path_type="absolute"
path="E:/OAE/database/NHTSA_SAMPLE"/>
</database_entries>
</global_config>
```

4.3 Preparation of CSV files

In order to leverage the batch programs bundled in OmniFind Analytics Edition that convert the source data to the desired Advanced Text Analytic Markup Language (ATML) format, the source data must be in Comma Separated Values (CSV) format.

Note: Exporting data to CSV files

If you have data in Microsoft Excel, use the **File** → **SaveAs** feature to save the data into CSV (Comma delimited) format.

If you have data in IBM DB2®, you can use DB2's export program to export your data into CSV format as follows:

```
db2 export to <csv file name> of del "select * from <database table name>"
```

For example, we want to export the data in table EMP_RESUME from the database SAMPLE and save it in resumes.csv. On Windows, following is a list of DB2 commands you can use to export the source data:

```
db2cmd
db2 connect to SAMPLE
db2 export to resumes.csv of del "select * from EMP_RESUME"
db2 terminate
```

Notice that **db2cmd** launches a DB2 command window.

If the source data is in CSV format, make sure that it conforms to the following rules:

- ▶ The separator for comma separated values is a single-byte comma (,).
 - ▶ All lines in the CSV file must have the same number of columns.
 - ▶ If line breaks or single-byte commas are used as part of data, these values must be enclosed in single-byte double quotation marks (").
- Special attention is required for non-text items (dates, for example) because these values are likely to be overlooked.
- ▶ When the values enclosed in double quotation marks contain double quotation marks as part of data, place two sets of double quotation marks.
- Example: The value "For example, "A" is an upper-case character." because the column value in the CSV file must be stored as follows:
- "For example, ""A"" is an upper-case character."
- ▶ The CSV file must be an appropriate text file.
 - ▶ Caution is necessary in handling binary data or character code conversion when converting data from other formats into CSV.

Strongly recommended rules:

- ▶ The number of lines in a CSV file must be 50,000 or less.
To handle data that contains more than 50,000 lines, we recommend dividing the data into multiple CSV files. The maximum number of lines depends on the size of data per line or on hardware specifications.
- ▶ Among CSV columns, identify the field that represents a date. The identified date field will be used in date-related search conditions, time-series analysis, and discovery. It is important to pick the right date field to be used.

If none of the columns is identified, the date of data conversion into the ATML format will be automatically attached to each line. If you generate ATML files today, then today's date will be attached to each ported document. Since this leads to no correlation between the attached date and data content, the result of time series analysis will be irrelevant.
- ▶ We recommend that you use the first line of the CSV file to list column names, and start describing the data on the second line.
Although this is not required, it will make later descriptions of correspondence between CSV and ATML easier.
- ▶ One CSV line must correspond to one document to be analyzed.
- ▶ Use the Microsoft Excel application to identify and remove redundant or duplicate data. For example, if multiple rows in the CSV file represents the same customer complaint, we recommend that you review and summarize all the duplicate complaints into one row.
- ▶ We also recommend verifying the validity of your CSV data file with the Microsoft Excel application. If you discover problems viewing the CSV data files with Microsoft Excel application, the OmniFind Analytics Edition batch program that converts the CSV files to ATML file will fail.

OmniFind Analytics Edition also provides a UIMA annotator that can be integrated with IBM OmniFind Enterprise Edition (OEE) to crawl and parse your enterprise data into the required MIML format. See Chapter 11, “Integrating OmniFind Analytics Edition with OmniFind Enterprise Edition” on page 295 for more details.

4.4 Database category creation

Categories in OmniFind Analytics Edition are representative label names assigned to keywords in the source data. Each keyword in the source data is assigned to at least one category name. For example, with the NHTSA sample database, we have assigned keywords such as *ignition*, *fuel tank*, and *engine* to a category called *Components*.

In the Text Miner application, the category names represent the keywords in the source data. For example, when the indexed data is sorted by category names, we can isolate documents that refer to *ignition*, *fuel tank*, or *engine* as the search results for the *Components category*.

Categories are divided into standard item category, system category, and user defined category. Their characteristics are summarized as follows:

- Standard item category:

Structured data within the source data can be categorized as the standard item categories. The source data used by OmniFind Analytics Edition contains multiple data entries (each representing a document to be analyzed). Each data entry contains a fixed number of columns, some of the columns represent structured data and some represent unstructured data (see Figure 4-4 for the columns in the NHSTA data).

The columns representing structured data can be selectively set up as categories, called the standard item categories in OmniFind Analytics Edition. If you think in terms of a relational database table, the structured column in the source data can be categorized, and the category name in OmniFind Analytics Edition is the column name in a database table. The category value in OmniFind Analytics Edition is the same as the column value in a database table. Not all columns in the source data become categories. You select only those that might be useful for analysis. For example, NHSTA data contains Manufacture Name, Vehicle Model, Date of Incident columns. They are structured data and are necessary for analysis. You can set them as categories, and they are considered *standard item categories*.

- System category:

Words from the unstructured data in the source data can be automatically categorized as system categories. OmniFind Analytics Edition provides natural language annotators that associates the unstructured content in the source data to Part of Speech (POS). For example in the NHSTA data, a column representing the description of complaints or incident is unstructured data. Part of a description might be "The door jammed." When the sentence is processed by the language annotators, the word *door* is assigned to a category called *noun*, the word *jammed* is assigned to a category called *verb*.

These types of categories, assigned by the language annotators from the unstructured content, are classified as *system categories*.

► User defined category:

From the unstructured content of the source data, users can assign special words, based on the data characteristics, as user defined categories. For example, in the NHSTA data, the description of complaints or incident (unstructured content) might contain words such as *door*, *ignition*, *fuel tank*, and *engine*. These words are not just simple nouns. They can provide useful insights for us during analysis related to the components of a vehicle. We therefore can categorize these special words as *Components*. These types of categories are called *user defined categories*. You assign these categories when using the Rules Editor or during the data ingestion process.

Categories can be further divided into subcategories. For example, category names such as noun, verb, or adjective are defined as subcategories for a category called *Part-of-speech*.

In OmniFind Analytics Edition, all the categories belong to a parent category called the *Root* category. The Root category is not available for representing any keywords and is also not editable. Figure 4-4 shows a diagrammatic representation of Root category, categories, and subcategories. The categories in OmniFind Analytics Edition represents a tree-like structure.

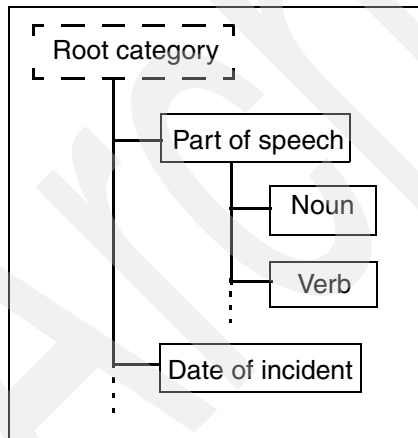


Figure 4-4 Category tree structure

After you have generated the source data in CSV format, you have to assign keywords in the columns of the source data to categories. Edit the `category_tree.xml` file to specify this keyword to category association. Also, edit the database settings in the `database_config.xml` file to identify the categories mentioned in `category_tree.xml` as standard item categories.

Category names marked as standard items in the database configuration file are not editable by the Dictionary Editor. For information on the dictionary, read Chapter 7, “Customizing the dictionary” on page 165.

In the sections to follow, we describe the steps to configure categories and databases. The category tree should be edited whenever new categories are added or the source data is modified.

Note: Any changes to the category tree requires a rebuild of the index.

4.4.1 Editing the category tree (category_tree.xml)

This section describes the association of CSV columns to category names and registering category names in category_tree.xml. The associated category names are used when the Data Ingestor converts the CSV files to ATML files.

The CSV data that we imported from NHTSA contains multiple rows of a database table with every column having a unique title. Table 4-1 defines the database schema of the NHTSA database.

Table 4-1 Schema for the sample NHTSA data

S No	Column Name	Data Type and description
1	CMPLID	CHAR(9) NHTSA's Internal Unique sequence number.
2	ODINO	CHAR(9) NHTSA's Internal reference number. This number can be repeated for multiple components.
3	MFR_NAME	CHAR(40) Manufacturer's name
4	MAKE	CHAR(25) Vehicle/equipment make
5	MODEL	CHAR(256) Vehicle/equipment model
6	YEAR	CHAR(4) Model year, 9999 if unknown or N/A
7	CRASH	CHAR(1) Was vehicle involved in a crash, 'Y' or 'N'
8	FAILDATE	CHAR(8) Date of incident (YYYYMMDD)

S No	Column Name	Data Type and description
9	FIRE	CHAR(1) Was vehicle involved in a fire 'Y' or 'N'
10	INJURED	NUMBER(2) Number of persons injured
11	DEATHS	NUMBER(2) Number of fatalities
12	COMPDESC	CHAR(128) Specific component's description
13	CITY	CHAR(30) Consumer's city
14	STATE	CHAR(2) Consumer's state
15	VIN	CHAR(11) Vehicle's VIN#
16	ADATE	CHAR(8) Date added to file (YYYYMMDD)
17	LDATE	CHAR(8) Date complaint received by NHTSA (YYYYMMDD)
18	MILES	NUMBER(6) Vehicle mileage at failure
19	OCCURENCES	NUMBER(4) Number of occurrences
20	CDESCR	CHAR(2048) Description of the complaint. Free text data that represents a complaint. This free text description will be broken down into nouns, adjectives, verbs, or other parts of speech by the language processing annotators in OmniFind Analytics Edition. For the rest of the chapter, we identify the contents of this column as unstructured source of data.

The first 19 columns of the NHTSA database represent structured data. Next we describe the steps required to associate the column values to category names.

The 20th column titled *CDESCR* represents a text string that has unstructured content and represents customer complaints. We describe the steps required to identify the 20th column as unstructured content so that the text strings in the 20th column can be processed by the language annotators to assign system categories (such as Part-of-speech, nouns, verbs, pronouns, or adjectives).

In OmniFind Analytics Edition, an entity for a category is made up of unique attributes called ID, path, and a name. The values for ID should be a unique positive numbers, path and name fields should be a unique string or a numerical numbers. In Table 4-2, we identify unique IDs, paths, and names for all the 19 structured data columns. In the rest of this section, we use the category names as set up in Table 4-2.

Table 4-2 Category path and name mapping

ID	Category path	Category name
200	CMPLID	CMPLID
201	ODINO	ODINO
202	MFR_NAME	MFR_NAME
203	MAKE	MAKE
204	MODEL	MODEL
205	YEAR	YEAR
206	CRASH	CRASH
207	FAILDATE	INCIDENT DATE
208	FIRE	FIRE
209	INJURED	INJURED
210	DEATHS	DEATHS
211	COMPDESC	COMPONENT DESCRIPTION
212	CITY	CITY
213	STATE	STATE
214	VIN	VIN
215	ADATE	ADD DATE
216	LDATE	RECEIVED DATE
217	MILES	MILES
218	OCCURENCES	OCCURENCES

After the unique category names are set up, you need to register these categories in a file called <DATABASE_DIR>\category\category_tree.xml. <DATABASE_DIR> represents the database directory.

Use a text editor to edit the category_tree.xml file, and add a node element to represent each category name. The highlighted text in Example 4-3 shows the additions to category_tree.xml.

Example 4-3 Sample category_tree.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<category_tree>
.....
.....
<node id="200" path="CMPLID" name="CMPLID" features=""/>
<node id="201" path="ODINO" name="ODINO" features=""/>
<node id="202" path="MFR_NAME" name="MFR_NAME" features=""/>
<node id="203" path="MAKE" name="MAKE" features=""/>
<node id="204" path="MODEL" name="MODEL" features=""/>
<node id="205" path="YEAR" name="YEAR" features=""/>
<node id="206" path="CRASH" name="CRASH" features=""/>
<node id="207" path="FAILDATE" name="INCIDENT DATE" features=""/>
<node id="208" path="FIRE" name="FIRE" features=""/>
<node id="209" path="INJURED" name="INJURED" features=""/>
<node id="210" path="DEATHS" name="DEATHS" features=""/>
<node id="211" path="COMPDESC" name="COMPONENT DESCRIPTION"
features=""/>
<node id="212" path="CITY" name="CITY" features=""/>
<node id="213" path="STATE" name="STATE" features=""/>
<node id="214" path="VIN" name="VIN" features=""/>
<node id="215" path="ADATE" name="ADD DATE" features=""/>
<node id="216" path="LDATE" name="RECEIVED DATE" features=""/>
<node id="217" path="MILES" name="MILES" features=""/>
<node id="218" path="OCCURENCES" name="OCCURENCES" features=""/>
.....
.....
</category_tree>
```

4.4.2 Editing database settings (database_config.xml)

The Dictionary Editor provided by OmniFind Analytics Edition can be used to create new categories and edit existing categories. In order to prevent modification to the categories as defined in 4.4.1, “Editing the category tree (category_tree.xml)” on page 65, it is mandatory to update the configuration file called <DATABASE_DIR>\conf\database_config.xml.

In the `database_config.xml` file, mark the categories we defined earlier as *reserved_by_system*. Categories marked as *reserved_by_system* cannot be edited by OmniFind Analytics Edition's Dictionary Editor.

To mark a category name as reserved by the system, an XML element called *category_entry* has to be created in `database_config.xml`. The values for the XML attributes called *name* and *value* should be specified as *reserved_by_system* and the user defined category name prepended with a period character. See the highlighted text in Example 4-4 for clarification.

Example 4-4 Entries defined in `database_config.xml`

```
<category_entries>
    <!-- Specifies subroot categories for system-reserved
categories. -->
    .....
    .....
    <category_entry name="reserved_by_system" value=".CMPLID"/>
        <category_entry name="reserved_by_system" value=".ODINO"/>
        <category_entry name="reserved_by_system" value=".MFR_NAME"/>
        <category_entry name="reserved_by_system" value=".MAKE"/>
        <category_entry name="reserved_by_system" value=".MODEL"/>
        <category_entry name="reserved_by_system" value=".YEAR"/>
        <category_entry name="reserved_by_system" value=".CRASH"/>
        <category_entry name="reserved_by_system" value=".FAILDATE"/>
        <category_entry name="reserved_by_system" value=".FIRE"/>
        <category_entry name="reserved_by_system" value=".INJURED"/>
        <category_entry name="reserved_by_system" value=".DEATHS"/>
        <category_entry name="reserved_by_system" value=".COMPDESC"/>
        <category_entry name="reserved_by_system" value=".CITY"/>
        <category_entry name="reserved_by_system" value=".STATE"/>
        <category_entry name="reserved_by_system" value=".VIN"/>
        <category_entry name="reserved_by_system" value=".ADATE"/>
        <category_entry name="reserved_by_system" value=".LDATE"/>
        <category_entry name="reserved_by_system" value=".MILES"/>
        <category_entry name="reserved_by_system" value=".OCCURENCES"/>
        <category_entry name="reserved_by_system" value=".CDESCR"/>
    .....
    .....
</category_entries>
```

The period in front of the category names prevents the category names from being displayed in the Dictionary Editor and hence they cannot be edited through the Dictionary Editor.

4.5 Generating ATML files

OmniFind Analytics Edition provides a Data Ingestor tool that converts a CSV file into an ATML file. ATML is a proprietary data format for OmniFind Analytics Edition. This section describes the procedures for executing the Data Ingestor command and related instructions on editing the Data Ingestor configuration file. The Data Ingestor configuration file maps the meaning for each data column in the source data.

Before running the Data Ingestor tool, it is mandatory to provide the CSV related column information in `<DATABASE_DIR>\category\category_tree.xml` and `<DATABASE_DIR>\conf\database_config.xml`.

4.5.1 Editing Data Ingestor configuration file (data_ingester_config_csv2atml.xml)

OmniFind Analytics Edition provides a batch process to convert the CSV data files into the proprietary ATML files. Before conversion takes place, the Data Ingestor configuration file called `data_ingester_config_csv2atml.xml` needs to be updated with CSV column information. In the configuration file, you identify the column fields in the CSV files, such as dates, titles, unique IDs, and columns that will be used for indexing.

After `data_ingester_config_csv2atml.xml` is updated, OmniFind Analytics Edition provides a batch process that uses the CSV file and the `data_ingester_config_csv2atml.xml` file to generate the ATML files required for further processing.

To explain how the `data_ingester_config_csv2atml.xml` file is updated, we use the sample NHTSA database as an example to explain the making of various fields and also describe the usage of the Data Ingestor batch process.

Note: Location of data_ingester_config_csv2atml.xml

The `data_ingester_config_csv2atml.xml` is located in `<DATABASE_DIR>\conf` directory. `<DATABASE_DIR>` is the file system path to your OmniFind Analytics Edition database.

In the configuration file, we must specify information for the following elements:

- ▶ `csv.character.encoding` (required)
- ▶ `csv.column.text.indexes` (required)
- ▶ `csv.row.firstindex` (required)
- ▶ `csv.column.names` (required)

- ▶ csv.date.format.list (optional)
- ▶ csv.column.index.id (optional)
- ▶ csv.column.index.date (optional)
- ▶ csv.date.index.title (optional)

csv.character.encoding (required)

This specifies the character coding for the input CSV file.

Setup example:

```
<param name="csv.character.encoding" multivalued="no">
<value>UTF-8</value>
</param>
```

In our example, we use UTF-8, which is the most used and recommended value. ASCII and MS932 are few of the other accepted values.

csv.column.text.indexes (required)

This specifies the non structured content columns (text columns) to be indexed by language processing to discover language related categories (Part-of-speech such as nouns, verbs, or adjectives).

Setup example:

```
<param name="csv.column.text.indexes" multivalued="yes">
<value>20</value>
<value>12</value>
</param>
```

In our example, we specify yes to multivalued field. The value 20 and 12 correspond to the column numbers 20 and 12 in the CSV file. The columns are related to *customer complaints* and *component descriptions*.

csv.row.firstindex (required)

This specifies the starting indexing column for porting CSV files to ATML files.

Setup example:

```
<param name="csv.row.firstindex" multivalued="no">
<value>1</value>
</param>
```

In our example, we specify column 1 in the CSV file as a starting column for porting CSV files to ATML files.

csv.column.names (required)

The entries in the ATML file that are ported from the corresponding columns in CSV file should be labeled. The label that is assigned from the corresponding CSV columns to the ATML file is represented by the unique category path names that we identified in the category_tree.xml file (see Example 4-3 on page 68).

In Example 4-5, we use the csv.column.names element to specify category path names for every entry that gets created in the ATML file.

Example 4-5 csv.column.names

```
<param name="csv.column.names" multivalued="yes">
<!--
Category path names should be prepended with a period. For example,
category path called CMPLID as identified in category_tree.xml is
entered as .CMPLID (notice the period in front of CMPLID).
-->
<value>.CMPLID</value>
<value>.ODINO</value>
<value>.MFR_NAME</value>
<value>.MAKE</value>
<value>.MODEL</value>
<value>.YEAR</value>
<value>.CRASH</value>
<value>.FAILDATE</value>
<value>.FIRE</value>
<value>.INJURED</value>
<value>.DEATHS</value>
<value>.COMPDESC</value>
<value>.CITY</value>
<value>.STATE</value>
<value>.VIN</value>
<value>.ADATE</value>
<!--
Column 17 is marked as field for csv.column.index.date, the
following category path .LDATE will be ignored by the Ingester and
replaced with a system default category path called .date
-->
<value>.LDATE</value>
<!--
Since we do not want to populate MILES information in the the ATML
files, we leave the category path for MILES as empty.
-->
<value></value>
<value>.OCCURENCES</value>
```



```

<!--
DESCRIPTION is not a category path in category_tree.xml
but column 20 is marked as csv.column.text.indexes. the ATML
Ingester will assign label called DESCRIPTION to the corresponding
CSV columns.
-->
<value>DESCRIPTION</value>
</param>

```

Following are the specifications for the XML elements listed in Example 4-5:

- ▶ *Always* be sure to set the same number of <value/> elements as the number of columns in the CSV file. In the NHTSA database, we have 20 columns and in Example 4-5, we create 20 <value/> elements.
- ▶ The data entered for the <value/> element should follow one of these rules:
 - Category path names should be prepended with a period. For example, category path called **CMPLID** as identified in `category_tree.xml` (see Example 4-3 on page 68) is entered as **.CMPLID** (notice the period in front of CMPLID in Example 4-5).
 - If you do not want language processing to process a column in the CSV file, the data for the corresponding <value/> element should be empty. In Example 4-5, we specify an empty value for the MILES column. Consequently, the MILES reported for each customer incident will not be ported to ATML and hence will not be indexed.
 - Category paths not beginning with a period can be assigned only to columns that are specified as `csv.column.text.indexes` fields. In Example 4-5, we specify a category path value of **DESCRIPTION** for the 20th column. Though **DESCRIPTION** is not a valid category path in `category_tree.xml` (Example 4-3), all the values for the 20th column in the CSV file will be labeled as **DESCRIPTION**. This feature in OmniFind Analytics Edition can be used to give user friendly names to column data.
 - Irrespective of the category path specified for the field mentioned as `csv.column.index.date`, during data porting to ATML file, the OmniFind Analytics Edition Data Ingester assigns a system category path called **.date** (notice the period in front of date). This is to let OmniFind Analytics Edition know the field that is used for date based analysis (Time series, 2D maps or any date based search conditions). In Example 4-5, column 17 is marked as field for `csv.column.index.date`, the assigned category path **.LDATE** will be ignored by the Data Ingester and replaced with a system default category path called **.date**.

csv.date.format.list (optional)

This specifies a list of possible date formats for the date values in the CSV files. The specified date format should be supported by the Java class `java.text.SimpleDateFormat`. If the format is not valid, then by default, the date when the ATML files are generated from CSV files will be assigned. This is an optional field. If it is not specified, then the default date pattern of “yyyyMMdd” will be used.

Setup example:

```
<param name="csv.date.format.list" multivalued="yes">
<value>yyyyMMdd</value>
<value>yyyy</value>
<value>yyyy-MM-dd</value>
<value>yyyy/MM/dd</value>
</param>
```

csv.column.index.id (optional)

This specifies the column number that provides the unique value for a document to be analyzed. This field is an optional field. If you do not specify the field, the IDs of the indexed documents will be marked sequentially.

Setup example:

```
<param name="csv.column.index.id" multivalued="no">
<value>1</value>
</param>
```

In our example, we specify column number 1 as the unique indexing column. The column represents the document ID that uniquely identify all the complain reports.

csv.column.index.date (optional)

This specifies the date field used to index the documents. Only one field can be specified as a date field. The identified date field will be used in the time series analysis and any search criteria that requires a date field. If your target data has more than one date field, it is crucial for you to understand the source data and choose a date field that provides more analysis value for your business. This field is an optional field. If you do not specify the date field, then by default, the date when you generate the ATML files from CSV files will be assigned.

Setup example:

```
<param name="csv.column.index.date" multivalued="no">
<value>17</value>
</param>
```

In our example, we specify column number 17 to represent the date of indexed document. Column 17 in the NHSTA CSV file specifies the *Date of incident*.

csv.date.index.title (optional)

This specifies the document title to be used for each data entry.

Setup example:

```
<param name="csv.date.index.title" multivalued="no">
<value>12</value>
```

In our example, we specify column number 1 to represent the document title.

Example of the Data Ingestor configuration file

Example 4-6 shows a sample copy of the data_ingester_config_csv2atml.xml file.

Example 4-6 Sample contents in data_ingester_config_csv2atml.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ingester_config SYSTEM "data_ingester_config.dtd">
<ingester_config>
....
<!--
=====
    Encoding of input CSV file.
    This parameter is mandatory.
=====
-->
<param name="csv.character.encoding" multivalued="no">
<value>UTF-8</value>
</param>
<!--
=====
    Index of the first row to be read.
    This parameter is mandatory.
=====
-->
<param name="csv.row.firstindex" multivalued="no">
<value>1</value>
</param>
<!--
=====
    Indexes of text columns.
=====
```

```

-->
<param name="csv.column.text.indexes" multivalued="yes">
<value>20</value>
<value>12</value>
</param>
<!--
=====
    Path or names for each column in csv.
=====
-->
<param name="csv.column.names" multivalued="yes">
<value>.CMPLID</value>
<value>.ODINO</value>
<value>.MFR_NAME</value>
<value>.MAKE</value>
<value>.MODEL</value>
<value>.YEAR</value>
<value>.CRASH</value>
<value>.FAILDATE</value>
<value>.FIRE</value>
<value>.INJURED</value>
<value>.DEATHS</value>
<value>.COMPDESC</value>
<value>.CITY</value>
<value>.STATE</value>
<value>.VIN</value>
<value>.ADATE</value>
<value>.LDATE</value>
<value>.MILES</value>
<value>.OCCURENCES</value>
<value>DESCRIPTION</value>
</param>
<!--
=====
    Formats for parsing date strings.
=====
-->
<param name="csv.date.format.list" multivalued="yes">
<value>yyyyMMdd</value>
<value>yyyy</value>
<value>yyyy-MM-dd</value>
<value>yyyy/MM/dd</value>
</param>
<!--

```

```

=====
    Index of a column for document id.
=====
-->
<param name="csv.column.index.id" multivalued="no">
<value>1</value>
</param>
<!--
=====
    Index of a column for date of - Date of incident.
=====
-->
<param name="csv.column.index.date" multivalued="no">
<value>17</value>
</param>
<!--
=====
    Index of a column for document title.
=====
-->
<param name="csv.column.index.title" multivalued="no">
<value>12</value>
</param>
....
</ingester_config>

```

4.5.2 Invoke Data Ingestor command to convert CSV to ATML format

As a prerequisite for generating ATML file from the CSV data file, it is necessary to finalize the category names in category_tree.xml and specify the column information in takmi_data_ingester configuration file.

Now you can launch a command window (on Windows) or shell (on AIX) to execute the ingester command as shown in Example 4-7.

Example 4-7 Usage of ingester command

Windows:

```
> takmi_data_ingester.bat <CONFIG_FILE> <CSV_FILE> <ATML_FILE>
```

AIX:

```
> takmi_data_ingester.sh <CONFIG_FILE> <CSV_FILE> <ATML_FILE>
```

The meanings of the arguments are as follows:

- ▶ <CONFIG_FILE>: Ingestor configuration file
- ▶ <CSV_FILE>: CSV file to be converted
- ▶ <ATML_FILE>: Path of the output ATML file to be generated. We recommend creating the output file in the <DATABASE_DIR>\db\atml directory. If hard disk space is a restriction, the generated output ATML files can be directed to a network file system.

Figure 4-5 shows how we executed the commands (in bold text) for our example and the execution output.

```
C:\OAE\database\NHTSA_SAMPLE>takmi_data_ingester.bat  
conf\data_ingester_config_csv2atml.xml  
c:\oae\database\NHTSA_SAMPLE\db\csv\NHTSA_SAMPLE.csv  
c:\oae\database\NHTSA_SAMPLE\db\atml\NHTSA_SAMPLE.atml  
Processing start.  
Processing completed for total 24 documents.  
C:\OAE\database\NHTSA_SAMPLE>
```

Figure 4-5 Output for *takmi_data_ingester.bat*

4.6 Natural language processing

Natural language processing is a crucial step in OmniFind Analytics Edition. The processing applies the language annotators, custom dictionary (if any), and custom rules (if any) to ATML files and generate corresponding MIML files.

Dictionary and rules can be edited through Dictionary Editor and Rules Editor. For more information, see Chapter 7, “Customizing the dictionary” on page 165 and Chapter 8, “Customizing rules” on page 189.

4.6.1 Allocating natural language processing resources

Before applying language processing to ATML files, it is necessary to update the database related configuration files with the list of language annotators or any custom defined dictionaries and rules. This preprocessing step involves the execution of the OmniFind Analytics Edition command (as shown in Example 4-8) and should be executed whenever new dictionaries or rules are created.

Launch a command window (on Windows) or shell (on AIX) to execute the command as shown in Example 4-8.

Example 4-8 Usage for takmi_nlp_resource_deploy commands

Windows:

> takmi_nlp_resource_deploy.bat <DATABASE_DIRECTORY>

AIX:

> takmi_nlp_resource_deploy.sh <DATABASE_DIRECTORY>

The meanings of the arguments are as follows:

- ▶ <DATABASE_DIRECTORY>: Path of the database directory.

Figure 4-6 shows how we executed the command (as shown in bold text) and the execution output.

```
C:\OAE\database\NHTSA_SAMPLE>takmi_nlp_resource_deploy.bat
c:\oae\database\NHTSA_SAMPLE
word boundary dictionary conversion
AnalyzerDictionaryConversion for C:\OAE\database\NHTSA_SAMPLE
Conversion completed
dictionary pattern generation
AnalyzerDictionaryPatternGenerator for C:\OAE\database\NHTSA_SAMPLE
Pattern generation completed
English mode [LW binary dictionary generation]
no user dictionary entries for the LW binary dictionary
C:\OAE\database\NHTSA_SAMPLE>
```

Figure 4-6 Output for takmi_nlp_resource_deploy.bat

4.6.2 Generating MIML files

OmniFind Analytics Edition provides a batch command to process the ATML files with the natural language annotators, dictionaries, and rules to generate MIML files. In addition to the source data, the generated MIML files have metadata discovered by the language annotators, custom dictionaries, and rules.

To invoke the language processing, launch a command window (on Windows) or shell (on AIX) and execute the command as shown in Example 4-9.

Example 4-9 Usage for takmi_nlp command

Windows:

```
> takmi_nlp.bat <DATABASE_DIRECTORY> <ATML_FILE> <MIML_FILE>
```

AIX:

```
> takmi_nlp.sh <DATABASE_DIRECTORY> <ATML_FILE> <MIML_FILE>
```

The meaning of the arguments are as follows:

- ▶ <DATABASE_DIRECTORY>: Absolute path of the database directory.
- ▶ <ATML_FILE>: Input ATML file name, including its *absolute file path*.
- ▶ <MIML_FILE>: Output MIML file name, including its *absolute file path*.

Note: You must provide the complete absolute paths for both the input ATML file and the output MIML file.

Figure 4-7 shows how we executed the commands (as shown in bold text) and the execution result.

```
C:\OAE\database\NHTSA_SAMPLE>takmi_nlp.bat c:\OAE\database\NHTSA_SAMPLE  
c:\OAE\database\NHTSA_SAMPLE\db\atml\NHTSA_SAMPLE.atml  
c:\OAE\database\NHTSA_SAMPLE\db\miml\NHTSA_SAMPLE.miml  
=== Initializing CollectionProcessingEngine  
InputFile: C:\OAE\database\NHTSA_SAMPLE\db\atml\NHTSA_SAMPLE.atml  
OutPutFile: C:\OAE\database\NHTSA_SAMPLE\db\miml\NHTSA_SAMPLE.miml  
=== Start Processing  
Processing completed  
TakmiPreprocessing: 3.109 sec.  
NumberOfDocuments: 24  
***** ABOUT LOG FILE *****  
* The location of the log file is specified by  
*   java.util.logging.FileHandler.pattern  
* property of  
*   C:\Program Files\IBM\takmi\conf\TAKMI_nlp_logging.properties  
* file.  
* The default location is C:\Program  
Files\IBM\takmi\logs\TAKMI_nlp_batch.log .  
*****  
C:\OAE\database\NHTSA_SAMPLE>
```

Figure 4-7 Sample output for takmi_nlp.bat

4.7 Indexing

With the source data transformed into MIML files, OmniFind Analytics Edition provides batch commands that will be used to generate the index. OmniFind Analytics Edition also provides batch commands to append new data to an existing index. In this section, we provide the instructions to create and update indexes.

4.7.1 Creating a new index

To build a new index from a set of MIML files, follow these steps:

1. Make sure that MIML files have been created through language processing, and that all the MIML files to be used for indexing are located in the `<DATABASE_DIRECTORY>\db\miml` directory.
2. Delete any old index as mentioned in 4.9, “Deleting data” on page 84. This step is required if you want to rebuild a new index.
3. Open a command window (on Windows) or shell (on AIX) to invoke the command shown in Example 4-10.

Example 4-10 Indexing command usage

Windows:

```
> takmi_index.bat <DATABASE_DIRECTORY> [HEAP_SIZE_MB]
```

AIX:

```
> takmi_index.sh <DATABASE_DIRECTORY> [HEAP_SIZE_MB]
```

The meaning of the arguments are as follows:

- ▶ `<DATABASE_DIRECTORY>`: Path of the database directory.
- ▶ `[HEAP_SIZE_MB]`: Optional. Java heap size when running commands. Specify in units of MBs. Default is 1000 MB. For a small set of data, the default value of 1000 MB is sufficient.

Figure 4-8 shows how we executed the command (as shown in bold text) and the execution output.

```

C:\OAE\database\NHTSA_SAMPLE>takmi_index.bat c:\OAE\database\NHTSA_SAMPLE
database_config.xml has been generated successfully.
***** ABOUT LOG FILE *****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern* property of
*   C:\Program Files\IBM\takmi\conf\tAKMI_indexer_logging.properties*
file.
* The default location is C:\Program
Files\IBM\takmi\logs\tAKMI_indexer_batch.log .
*****
DataEntry[db\miml\NHTSA_SAMPLE.miml] processing start.
***
DataEntry[db\miml\NHTSA_SAMPLE.miml] processing done.
***** ABOUT LOG FILE *****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern* property of
*   C:\Program Files\IBM\takmi\conf\tAKMI_indexer_logging.properties*
file.
* The default location is C:\Program
Files\IBM\takmi\logs\tAKMI_indexer_batch.log .
*****
File Merge start.
***** ABOUT LOG FILE *****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern* property of
*   C:\Program Files\IBM\takmi\conf\tAKMI_indexer_logging.properties*
file.
* The default location is C:\Program
Files\IBM\takmi\logs\tAKMI_indexer_batch.log .
*****
File Merge done.
Group Merge start.
.....
***** ABOUT LOG FILE *****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern* property of
*   C:\Program Files\IBM\takmi\conf\tAKMI_indexer_logging.properties*
file.
* The default location is C:\Program
Files\IBM\takmi\logs\tAKMI_indexer_batch.log .
*****
Group Merge done.
Index Build done.

C:\OAE\database\NHTSA_SAMPLE>

```

Figure 4-8 Sample output for takmi_index.bat

4.7.2 Updating an index by adding files

If any new data is added to the source data, OmniFind Analytics Edition provides batch commands to update an existing index. To update an existing index, follow these steps:

1. Prepare a new set of MIML files that contain a new set of processed information. In our example, we created a new MIML file called `nhtsa2.miml`, which is created through language processing and it is located in `<DATABASE_DIRECTORY>\db\miml` directory.
2. Open a command window (on Windows) or shell (on AIX) to invoke the command as shown in Example 4-11.

Example 4-11 Usage of index update build command

Windows:

```
> takmi_index_diff.bat <DATABASE_DIRECTORY> [HEAP_SIZE_MB]
```

AIX:

```
> takmi_index_diff.sh <DATABASE_DIRECTORY> [HEAP_SIZE_MB]
```

The meanings of the arguments are as follows:

- ▶ `<DATABASE_DIRECTORY>`: Path of the database directory.
- ▶ `[HEAP_SIZE_MB]`: Optional. Java heap size when running commands. Specify in units of MBs. Default is 1000 MB. For a small set of data, the default value of 1000 MB is sufficient.

When `takmi_index_diff.bat` is invoked, it checks for new MIML files in `<DATABASE_DIRECTORY>\db\miml` and appends to the existing index database.

4.8 Accessing Text Miner application

Text Miner is a J2EE application bundled with OmniFind Analytics Edition to provide the analytic tools required to navigate through data. This section describes how to access Text Miner application and validate the index.

As a prerequisite to access the index with Text Miner, it is necessary to register the database in `global_config.xml` and recycle the Text Miner application.

Refer to section 4.2, “Database directory creation and registration” on page 58 to confirm the additions to `global_config.xml`. With the global changes in effect, access the index by entering the following URL in a browser:

`http://<Server Name:Port Number>/TAKMI_MINER`

In this URL, the <Server Name> is where you installed the Text Miner application and the port number by default is 9080.

Figure 4-9 shows the Welcome page of the Text Miner application, and also displays the new database (NHTSA_SAMPLE). Now click the **NHTSA_SAMPLE** database link to confirm the validity of the new index.

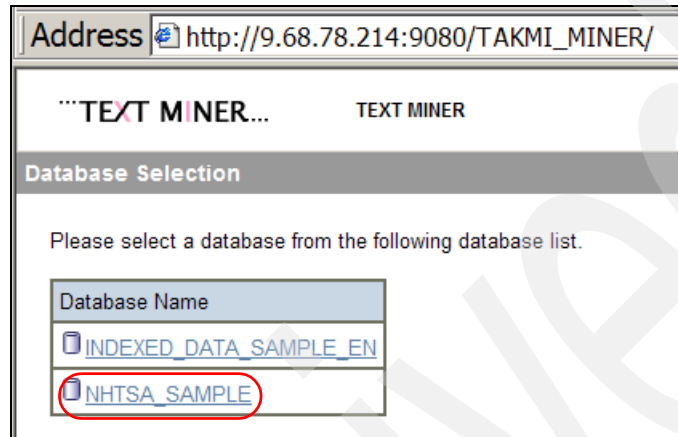


Figure 4-9 Text Miner interface to select database

4.9 Deleting data

OmniFind Analytics Edition also provides batch commands to delete index and MIML files.

Note: When to stop the Websphere application server

To delete the index or MIML files, it is mandatory to stop the WebSphere Application Server.

If the source data is updated with new additions or existing data is modified, it is mandatory to delete the existing index and rebuild the index from a new set of MIML files. The action prevents any duplicate data from going into the index.

In Example 4-12, we display the OmniFind Analytics Edition command that deletes the index files for a specified database.

Example 4-12 Usage of the command to clear index database

Windows:

```
> takmi_clear_index.bat <DATABASE_DIRECTORY>
```

AIX:

```
> takmi_clear_index.sh <DATABASE_DIRECTORY>
```

When new dictionary or rules or categories are applied to the source data, the preprocessing results in new sets of MIML files and index. In such scenarios, before the preprocessing steps are performed on the source data, you should delete the existing index and MIMIL files.

In Example 4-13, we display the OmniFind Analytics Edition command that deletes the MIML and index files for a specified database.

Example 4-13 Usage of the command to clear the NLP index

Windows:

```
> takmi_clear_nlp_index.bat <DATABASE_DIRECTORY>
```

AIX:

```
> takmi_clear_nlp_index.sh <DATABASE_DIRECTORY>
```

Figure 4-10 shows how to execute the command (as shown in bold text) and the execution result.

```
C:\OAE\database\NHTSA_SAMPLE>takmi_clear_nlp_index.bat
c:\OAE\database\NHTSA_SAMPLE
***** WARNING *****
* THIS COMMAND CLEARS ALL MIML FILES *
* AND INDEX FILES. *
*****
OK? Please enter Y or N :y
directory "c:\OAE\database\NHTSA_SAMPLE\db\index" and all miml files in
"c:\OAE\
database\NHTSA_SAMPLE\db\miml" were successfully deleted.

C:\OAE\database\NHTSA_SAMPLE>
```

Figure 4-10 Sample output for takmi_clear_nlp_index.bat

As for preprocessed ATML files, you can regenerate the ATML files if new source data is provided in the CSV files. After the index is built, the ATML files can be manually deleted to restore hard disk space.

4.10 Importing data using OmniFind Enterprise Edition

OmniFind Enterprise Edition (OEE) provides a variety of crawlers to extract content from your enterprise repositories. The crawlers in OEE can be leveraged to ingest data into OmniFind Analytics Edition (thus bypassing the CSV file generation and ATML file conversion process and creating MIML files directly). Refer to Chapter 11, “Integrating OmniFind Analytics Edition with OmniFind Enterprise Edition” on page 295 for more details.

4.11 Quick reference for data ingestion and indexing

We summarize the steps, the associated files to be updated, and commands for data ingestion and indexing for quick reference in this section (Figure 4-11):

1. Create database directory and register a database name.
Update `global_config.xml` with the new database name.
2. Export data into CSV format.
3. Identify categories and configuration database.
Update `category_tree.xml` with categories.
Update `database_config.xml` with category names as reserved_by_system.
4. Convert CSV to ATML using Data Ingestor:
Update `data_ingester_config_csv2atml.xml` with mapping information.
`takmi_data_ingester.bat <CONFIG_FILE> <CSV_FILE> <ATML_FILE>`
5. Convert ATML to MIML using OAE preprocessor (NLP):
`takmi_nlp_resource_deploy.bat <DATABASE_DIRECTORY>`
`takmi_nlp.bat <DATABASE_DIRECTORY> <ATML_FILE> <MIML_FILE>`
6. Index MIML files using Indexer:
`takmi_index.bat <DATABASE_DIRECTORY> [HEAP_SIZE_MB]`

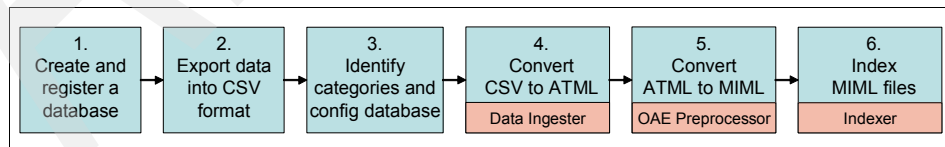


Figure 4-11 Data ingestion and indexing steps

We provide Windows commands in the quick reference only. For AIX commands, use `.sh` instead of `.bat`.

Text Miner mining basics

IBM OmniFind Analytics Edition (OAE) provides a standalone J2EE application called Text Miner that can help analyze the indexed source dataset. The analysis leads to identifying leading trends, early detection of problems, or understanding the voice of customers, which in turn can help grow your business.

In this chapter we explore the navigational features of the Text Miner application. Throughout the chapter, we use the examples from a sample database we have built with the complaints dataset downloaded from the National Highway Traffic Safety Administration (NHTSA).

We cover the following topics:

- ▶ Text Miner
- ▶ Category tree view
- ▶ Search
- ▶ Views
- ▶ Reporting features

5.1 Text Miner

Text Miner is a J2EE Web application that is provided as an out-of-the-box application. Text Miner provides facilities for real-time statistical analysis on the index for a source dataset. It allows the users to analyze the processed data by organizing the data into categories, applying search conditions, and further drilling down to analyze data from different perspectives such as over a period of time, changes in data, changes in topic, or as a matrix of two categories. Text Miner is the application that will help identify useful trends and predominant patterns hidden in the processed data.

OmniFind Analytics Edition also comes with a wide set of Java APIs that can be used to develop custom real-time analyzing applications. Refer to the product manuals for information about the mining APIs.

Notes:

- ▶ Throughout this chapter, we explain the features of Text Miner application by navigating the index built for the sample complaints database published by National Highway Traffic Safety Administration (NHTSA).
- ▶ At the time of writing this book, the J2EE applications bundled with OmniFind Analytics Edition Version 8.4 are supported officially only on Microsoft Internet Explorer Version 6.x.

5.1.1 Deploying and accessing the application

OmniFind Analytics Edition provides a deployment script tailored for IBM WebSphere Application server Version 6.x. The script deploys the Text Miner application as one of the four applications bundled in the product. Refer to 3.4, “Verification of installation and configuration” on page 50 for in-depth details on installation.

When the installation is successful, the Text Miner application can be accessed at:

`http://<Host Name:Port Number>/TAKMI_MINER`

Host name is the server where you deployed the Text Miner application. Port number, by default, is 9080. Refer to 4.8, “Accessing Text Miner application” on page 83 for details on accessing the first page of the application.

5.1.2 Window layout and functional overview

The Text Miner interface is separated into four areas as shown in Figure 5-1.

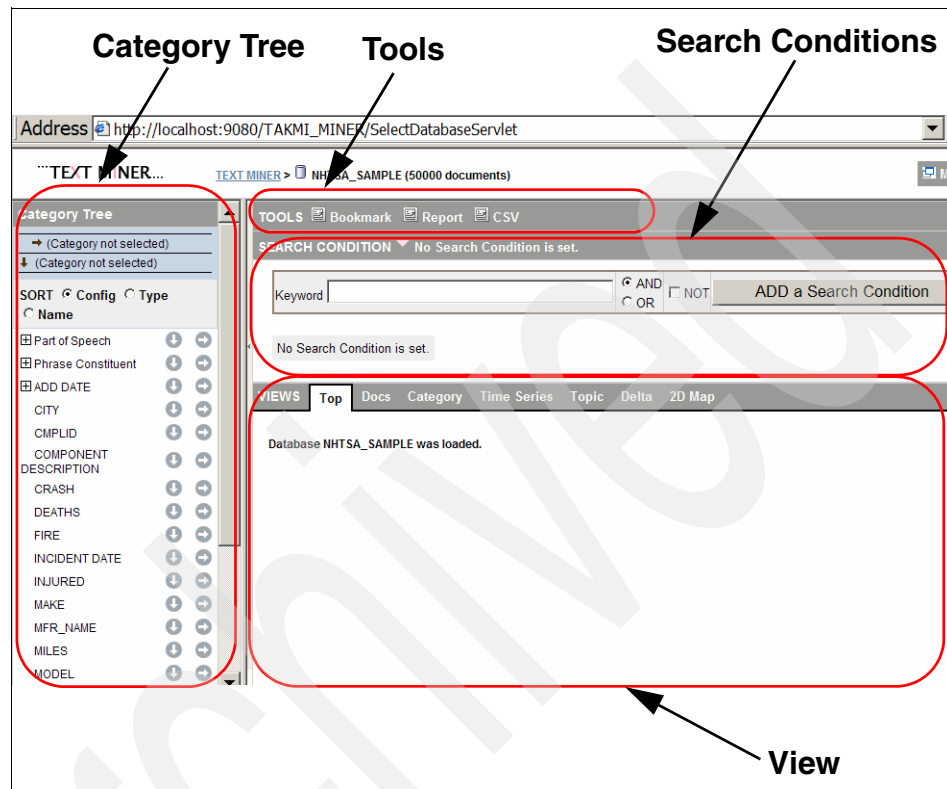


Figure 5-1 Window layout of Text Miner

Four areas of the Text Miner application are:

- ▶ **Category tree:** This is where you select the categories to be analyzed.
- ▶ **Tools:** This area shows the tools (or functions) that are available in the interface. For example, you can export results from an analysis.
- ▶ **Search conditions:** This is where you create search conditions by specifying keywords and selecting search operators (AND, OR and NOT).
- ▶ **View:** You can analyze the retrieved documents narrowed down by the search conditions in various views. Text miner provides seven different views that can be used to display the search results.

In the following sections of this chapter we elaborate more on each of these areas.

5.1.3 Database selection window

Text Miner application supports analysis of multiple databases. A database has to be selected from a list of databases that are configured in the file, *global_config.xml*. A single instance of Text Miner can analyze only one selected instance of the database. Multiple instances of Text Miner applications can be launched to analyze other databases. When the Text Miner is first launched in a browser, the Welcome page (Figure 5-2) prompts you to select a database to continue.

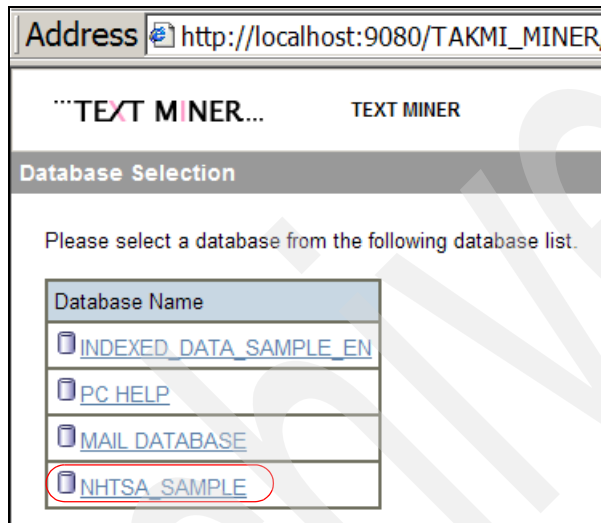


Figure 5-2 Database selection page

In the sections to follow, we use the database titled NHTSA_SAMPLE for understanding the Text Miner application. It is the complaints dataset, available for public viewing, that can be downloaded from the National Highway Traffic Safety Administration (NHTSA).

5.2 Category tree view

Category is a label name given to keywords detected in source data. In Figure 5-3 we display the categories discovered by the OmniFind Analytics Edition language processors from the unstructured contents and the categories assigned from the structured content of the source data.

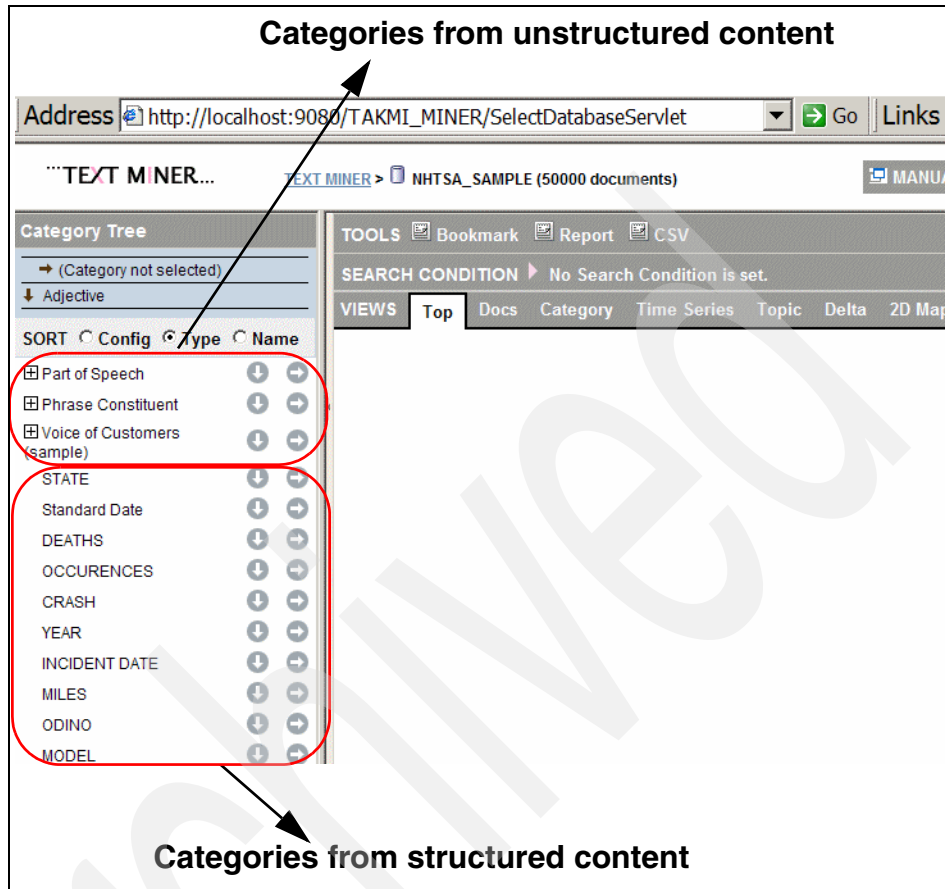


Figure 5-3 Category view

Refer to 4.4.1, “Editing the category tree (category_tree.xml)” on page 65 where we describe the schema of the sample NHTSA data.

For this database, each data entry represents a complaint record from someone. The first 19 columns of each record contain unique features of the involved vehicles and the incident, for example, model of the vehicle, miles that the car has driven, the manufacturer name, the make and model of the car, and whether there is a fire or death involved in the incident. This data is structured data. The 20th column of the complaint record represents complaints written by users in free text. This information is considered unstructured content, because it is free formatted and you have no idea what the users have written.

When we apply OmniFind Analytics Edition's preprocessing functions to NHTSA data, the keywords in the first 19 columns are assigned to the column names, and these column names are represented as categories in the Text Miner application. In Figure 5-3, these categories are highlighted as *categories from structured content*.

The OmniFind Analytics Edition natural language annotators analyzes the free formatted text in column 20 and assigns keywords to system categories and subcategories.

To be precise, the free text gets assigned to system categories called:

- ▶ Part of Speech, with subcategories such as adjectives, nouns, and verbs.
- ▶ Phrase Constituent, with subcategories such as noun sequence and noun verb.
- ▶ Voice of Customers, with subcategories such as unfavorable, gratitude, and questions.

In Figure 5-3, these categories are highlighted as *categories from unstructured content*.

5.2.1 Sorting categories

The categories list in Figure 5-3 can be sorted by Config, Type, or Name. Figure 5-4 shows the radio buttons that can be used to sort the displayed category names.

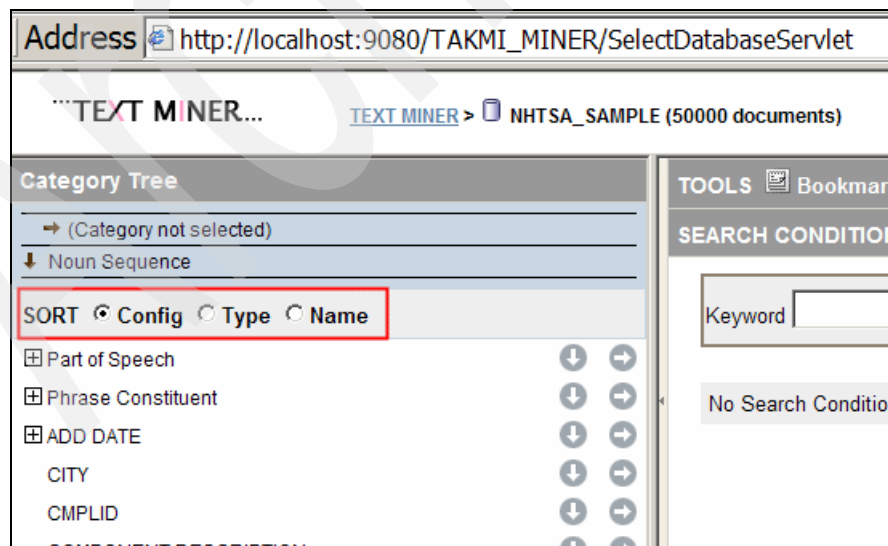


Figure 5-4 Sort radio buttons for category names

Following is a description of each radio button for sorting:

- **Config:** The categories are listed in order of creation in the configuration (database_config.xml) file. The categories identified by the XML element called <frequently_used_category_entries/> are displayed at the top, followed by the alphabetical sorted categories identified from the structured data and finally the categories identified by the XML element called <infrequently_used_category_entries/>.
- **Type:** The categories with subcategories are listed in alphabetical order.
- **Name:** The categories are listed in alphabetical order.

5.2.2 Display and hide subcategories

When the category has a subcategory or subcategories, a button is displayed in front of the category name so that the subcategory and subcategories can be displayed or hidden. Figure 5-5 shows a sample view where the categories can be displayed by clicking [+] and hidden by clicking [-].

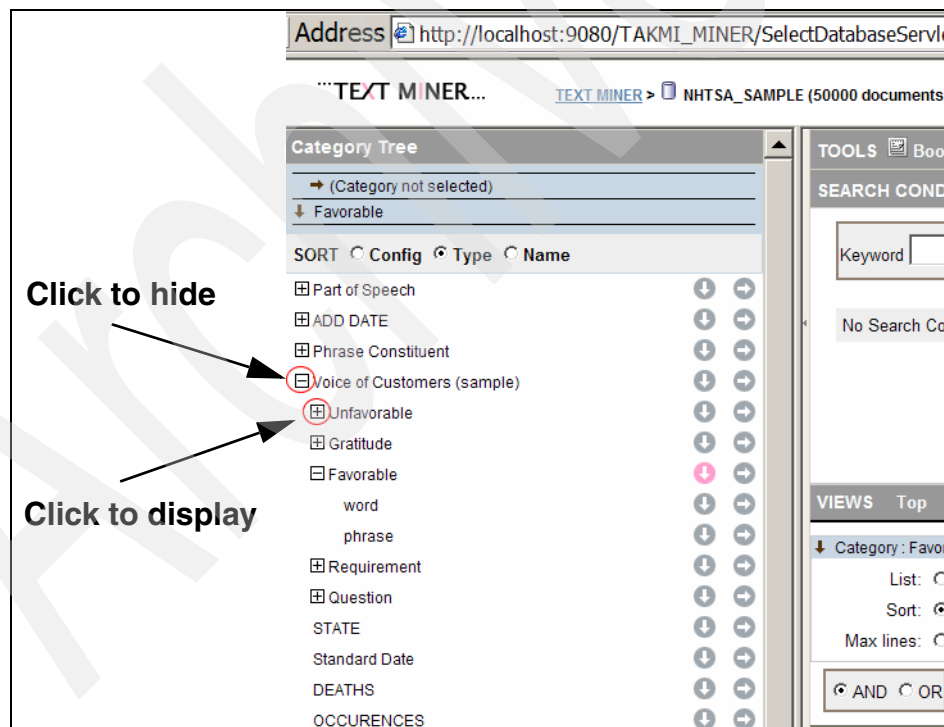


Figure 5-5 Display or hide subcategories

5.2.3 Selecting categories and views

Categories can be selected by using the vertical and horizontal arrow buttons of the Category view.

Selecting categories and views:

- ▶ Arrows marked in GREY color means that they are unselected.
- ▶ Arrows marked in PINK color means that they are selected.
- ▶ A selected PINK colored arrow can be unselected by clicking it. The arrow color would change back to GREY.
- ▶ Categories are not used as parameters in Docs view and Time Series view.
- ▶ A selected category is listed on the vertical axis of Category view, Topic view, and Delta view.
- ▶ Selected categories are listed on both the vertical and horizontal axis of a 2D Map.

To illustrate the Category view, refer to Figure 5-6, where we display the Category view when the subcategory called **Noun Sequence** is selected.

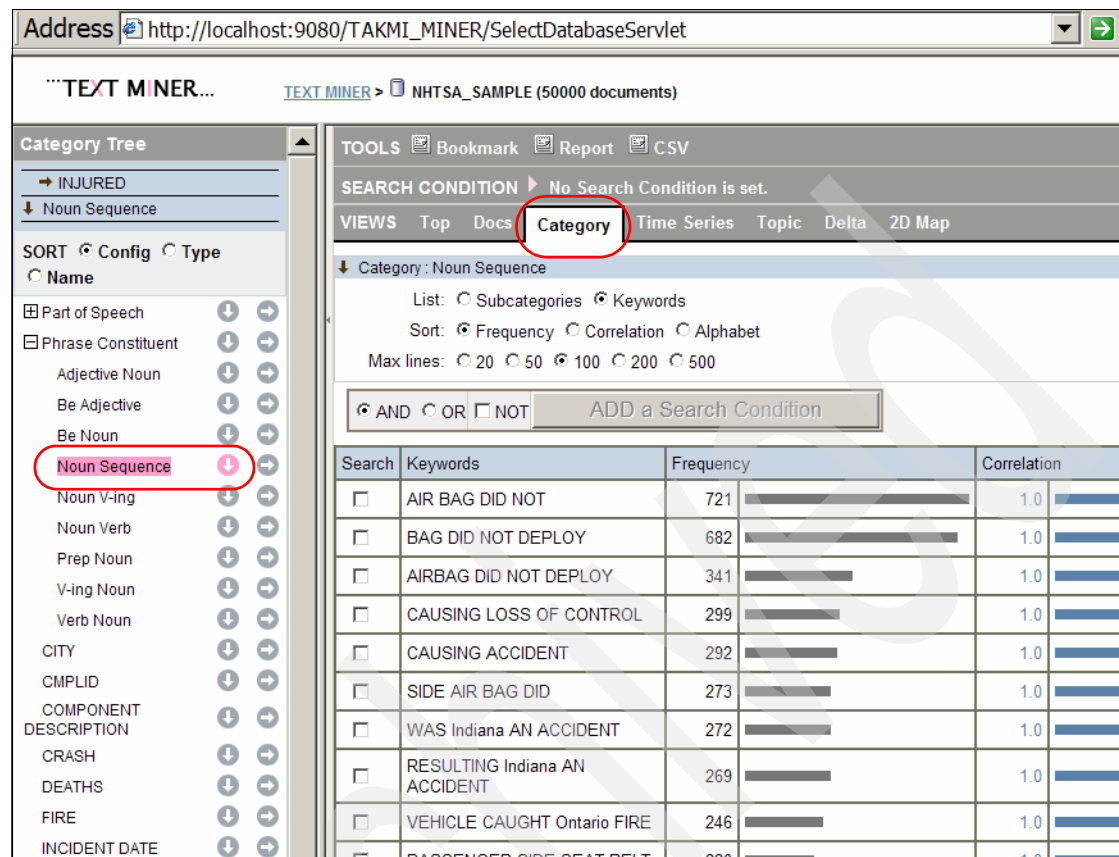


Figure 5-6 Category view for Noun Sequence

In Figure 5-6, it shows a list of noun-sequences and how often they appear in the source data. For example, the noun-sequence, “AIR BAG DID NOT” occurred 721 times in the source data. We discuss more about this view in 5.4.3, “Category view” on page 116.

Next, in Figure 5-7, we display the 2D Map view when the vertical arrow for subcategory called Noun Sequence is selected and the horizontal arrow for category called CRASH is selected.

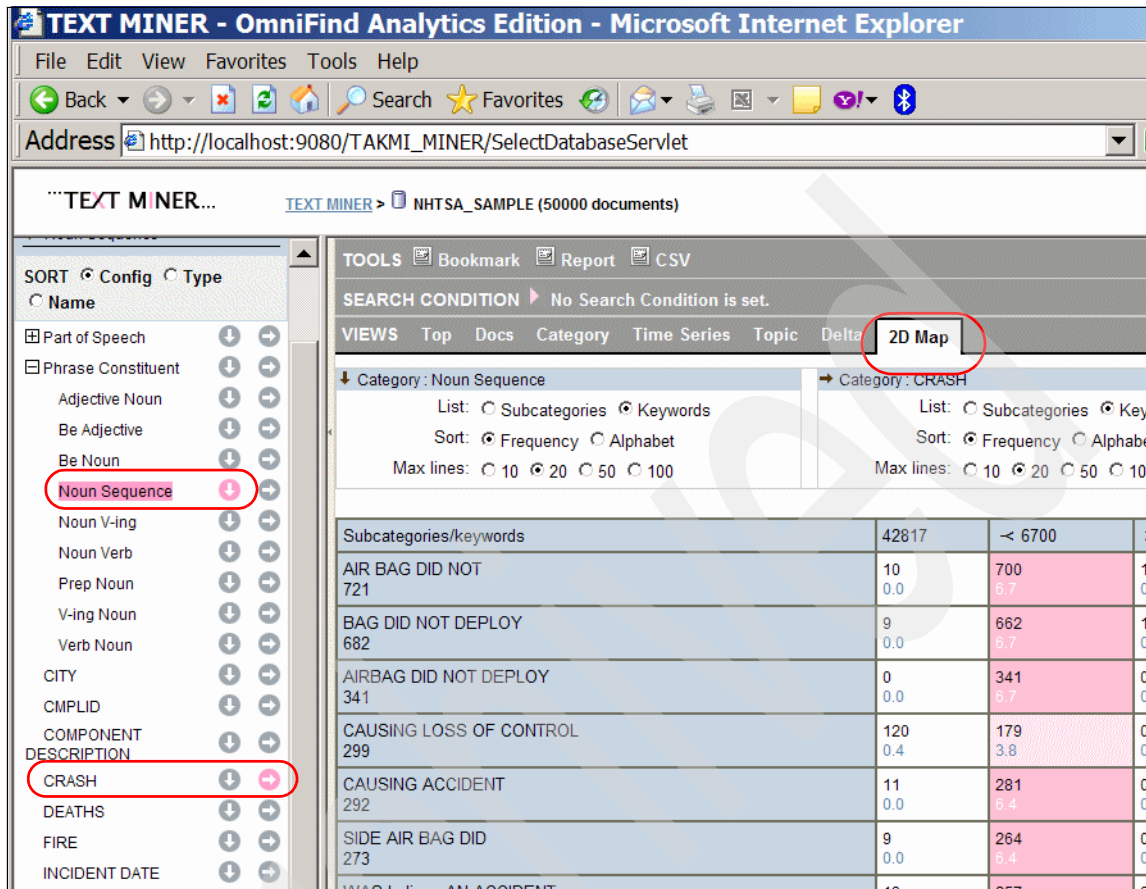


Figure 5-7 2D view for Noun Sequence and CRASH

In Figure 5-7, the 2D Map view shows the noun-sequences (such as “AIR BAG DID NOT”) that are associated with vehicle crash. We will discuss more about the 2D view in 5.4.7, “2D Map view” on page 128.

5.3 Search

In the Text Miner application, search conditions can help to filter out the documents. Search conditions aid in a focused analysis of the source data.

OmniFind Analytics Edition provides the following types of search conditions:

- **Keyword search:** Documents in the database are narrowed down by the keywords found in the dictionaries.

- ▶ **Category search:** Documents in the database are narrowed down by keywords associated to a particular category. The association is applied during the data ingestion process.
- ▶ **Date search:** Documents in the database are narrowed down by a date criteria.

OmniFind Analytics Edition also provides AND, OR, and NOT operators to combine any of the above mentioned search conditions.

For the rest of this section, we provide in-depth details on the Text Miner user interface and the search conditions.

5.3.1 Keyword search

In keyword search, you enter the keywords that need to be searched among the documents. These keywords are registered in the OmniFind Analytics Edition system dictionaries and the user defined dictionaries. System dictionaries associate keywords to natural language grammar such as Nouns, Verbs, Adverbs, and Adjectives. User dictionaries are created by users to define new category names with specific keywords and synonyms.

When synonyms are registered as keywords, documents that contain synonyms of the keywords are also retrieved. For example, if the word “DUI” is set as a synonym for the keyword “driving under influence,” a keyword search on “driving under influence” will also return documents containing the synonym keyword “DUI”. Refer to Chapter 7, “Customizing the dictionary” on page 165 to learn about creating synonyms with Dictionary Editor.

Keyword search and views

In Text Miner, you have to select the category to be displayed in the Category, Topic, Delta, and 2D Map views. But for the Doc and Time Series views, you do not need to select a category from the category tree.

With an example, we provide the sequential instructions required to display a keyword search results in a category view.

As an example, we try to identify the complaints where the keyword fire was related to a vehicle component called FUEL SYSTEM: GASOLINE.

1. Launch the Text Miner application and select *NHTSA_SAMPLE* database.
2. As shown in Figure 5-8, enter the keyword fire in the input text box and click the **ADD a Search Condition** button.

SEARCH CONDITION ▼ Searched documents: 94 documents

Keyword ☒ AND ☐ OR ☐ NOT ADD a Search Condition

Figure 5-8 Simple search

- In Figure 5-9, we select a category called *COMPONENT DESCRIPTION* and list the search results in the Category view.

Address

TEXT MINER... [TEXT MINER](#) > NHTSA_SAMPLE (50000 documents)

Category Tree

- (Category not selected)
- COMPONENT DESCRIPTION

SORT ☒ Config ☐ Type

☐ Name

- Part of Speech
- Phrase Constituent
- CITY
- CMPLID
- COMPONENT DESCRIPTION
- CRASH
- DEATHS
- FIRE
- INCIDENT DATE
- INJURED
- MAKE
- MFR_NAME
- MILES
- MODEL
- OCCURENCES
- ODINO
- RECEIVED DATE
- STATE

TOOLS [Bookmark](#) [Report](#) [CSV](#)

SEARCH CONDITION ▼ Searched documents: 94 documents

Keyword ☒ AND ☐ OR ☐ NOT ADD a Search

Searched documents: 94 documents

KEYWORD: fire 94 docs [DELETE](#)

VIEWS [Top](#) [Docs](#) Category [Time Series](#) [Topic](#) [Delta](#) [2D Map](#)

Category: COMPONENT DESCRIPTION

List: ☐ Subcategories ☒ Keywords

Sort: ☒ Frequency ☐ Correlation ☐ Alphabet

Max lines: ☐ 20 ☐ 50 ☒ 100 ☐ 200 ☐ 500

☒ AND ☐ OR ☐ NOT ADD a Search Condition

Search	Keywords	Frequency	Correlation
<input type="checkbox"/>	ENGINE AND ENGINE COOLING:ENGINE:GASOLINE	63	22.0
<input checked="" type="checkbox"/>	FUEL SYSTEM, GASOLINE	6	2.6
<input type="checkbox"/>	ELECTRICAL SYSTEM:IGNITION:SWITCH	6	1.0
<input type="checkbox"/>	ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD	4	1.1

Figure 5-9 Category view for fire and listed by component description

- Figure 5-9 shows 94 complaints referring to the keyword fire. From this set of complaints, we want to identify the number of complaints that are related to FUEL SYSTEM: GASOLINE. In order to append the two keywords, we select FUEL SYSTEM: GASOLINE, select the **AND** radio button and then click **ADD a Search Condition**. Refer to Figure 5-10 for the appropriate display.

VIEWS Top Docs **Category** Time Series Topic Delta 2D Map

Category: COMPONENT DESCRIPTION

List: ☐ Subcategories ☒ Keywords

Sort: ☒ Frequency ☐ Correlation ☐ Alphabet

Max lines: ☐ 20 ☐ 50 ☒ 100 ☐ 200 ☐ 500

☒ AND ☐ OR ☐ NOT **ADD a Search Condition** ②

Search	Keywords	Frequency
<input type="checkbox"/>	ENGINE AND ENGINE COOLING:ENGINE:GASOLINE	63
<input checked="" type="checkbox"/> ①	FUEL SYSTEM, GASOLINE	6

Figure 5-10 AND condition on multiple keywords

5. In Figure 5-11, we display the search results of the AND condition.

FUEL SYSTEM is appended as AND condition

SEARCH CONDITION Searched documents: 6 documents

Keyword ☒ AND ☐ OR ☐ NOT **ADD a Search Condition**

Searched documents: 6 documents

AND 6 docs [DELETE](#)

KEYWORD: fire 94 docs

KEYWORD: FUEL SYSTEM, GASOLINE(COMPONENT DESCRIPTION) 345 docs

VIEWS Top Docs **Category** Time Series Topic Delta 2D Map

Category: COMPONENT DESCRIPTION

List: ☐ Subcategories ☒ Keywords

Sort: ☒ Frequency ☐ Correlation ☐ Alphabet

Max lines: ☐ 20 ☐ 50 ☒ 100 ☐ 200 ☐ 500

☒ AND ☐ OR ☐ NOT **ADD a Search Condition**

Search	Keywords	Frequency
<input type="checkbox"/>	FUEL SYSTEM, GASOLINE	6

Figure 5-11 Search results with AND condition

Understanding the search results:

In Figure 5-11, we observe:

- ▶ There are 94 documents that contain the keyword fire (in the text content).
- ▶ There are 345 documents have FUEL SYSTEM. GASOLINE as the value for the category called COMPONENT DESCRIPTION.
- ▶ When the keywords fire and FUEL SYSTEM. GASOLINE are combined with an AND operator, we discover 6 complaints where fire was referenced with FUEL SYSTEM. GASOLINE.

Figure 5-12 displays a sample document from the search results we received from the query executed displayed in Figure 5-11. In Figure 5-12, we have highlighted the keywords that were searched.

Document Inspector	
Document Information	
Document ID	8702
Title	8702
Text	DESCRIPTION: VEHICLE FIRE EXPLOSION, CAUSED BY POSSIBLE LEAK OF GASOLINE. *SKD
Standard Information	
Category	Keyword
ADD DATE	19950203
CITY	LEBANON
CMPLID	8702
COMPONENT DESCRIPTION	FUEL SYSTEM, GASOLINE
CRASH	
DEATHS	0
FIRE	Y
INCIDENT DATE	19940314
INJURED	0

Figure 5-12 sample document identified by the search results

Expectations on keyword search results

Keywords that are used in keyword search are not just simple character strings, but instead, they are the keywords extracted from the target documents by natural language processing. For this reason, while it is possible to retrieve documents that contain synonyms, there might be some cases in which documents that contain character strings including the text to be searched might not be retrieved.

For example, documents containing a phrase “automatic transmission” will not be retrieved if you run a keyword search with the term “automatic.” This is because the phrase “automatic transmission” is recognized as one word by the language processing, and a character string “automatic” contained in that phrase is not recognized as a word. This is an advantage of OmniFind Analytics Edition where you can customize it to understand the entire phrase “automatic transmission” as a complete word. You can then discover problems or issues associated with “automatic transmission” — for example, for a particular vehicle make or model.

In keyword search, the category to which the keyword belongs affects the search result. This is because OmniFind Analytics Edition differentiates keywords in a different context (that is, category). Therefore, even for the same keyword, the search result might be different depending on which Category view or which text field the search is performed from.

For example, assume there is a dictionary that contains the following information:

- ▶ AC (keyword) = Air Conditioner (synonym) is registered in the category called “Component.”
- ▶ AC (keyword) = Alternate Current (synonym) is registered in the category called “Power Specifications”.

The Component category represents vehicle components. When you search for “AC” in the Component category, OmniFind Analytics Edition will return the documents that contain “AC” or “Air Conditioner,” which are part of the vehicle components (because they are within the Component category). When you restrict the category to the Power Specifications and perform a search on the same search word “AC”, then only the documents that contain the word “AC” or “Alternate Current” within the Power Specifications category will be retrieved.

5.3.2 Category search

When performing a category search, the search is based on keywords that are registered in the database and are associated with a particular category or subcategory.

You do not have to enter any search keywords for a category search. In Figure 5-13, we show the Category view by simply selecting a category from the category tree. When a category is selected from the category tree and the Category view is chosen for display, a search condition is executed to list the keywords assigned to the selected category across all the documents in the dataset. In Figure 5-13, we display the keywords across all the 50000 documents that are associated to the category called COMPONENT DESCRIPTION.

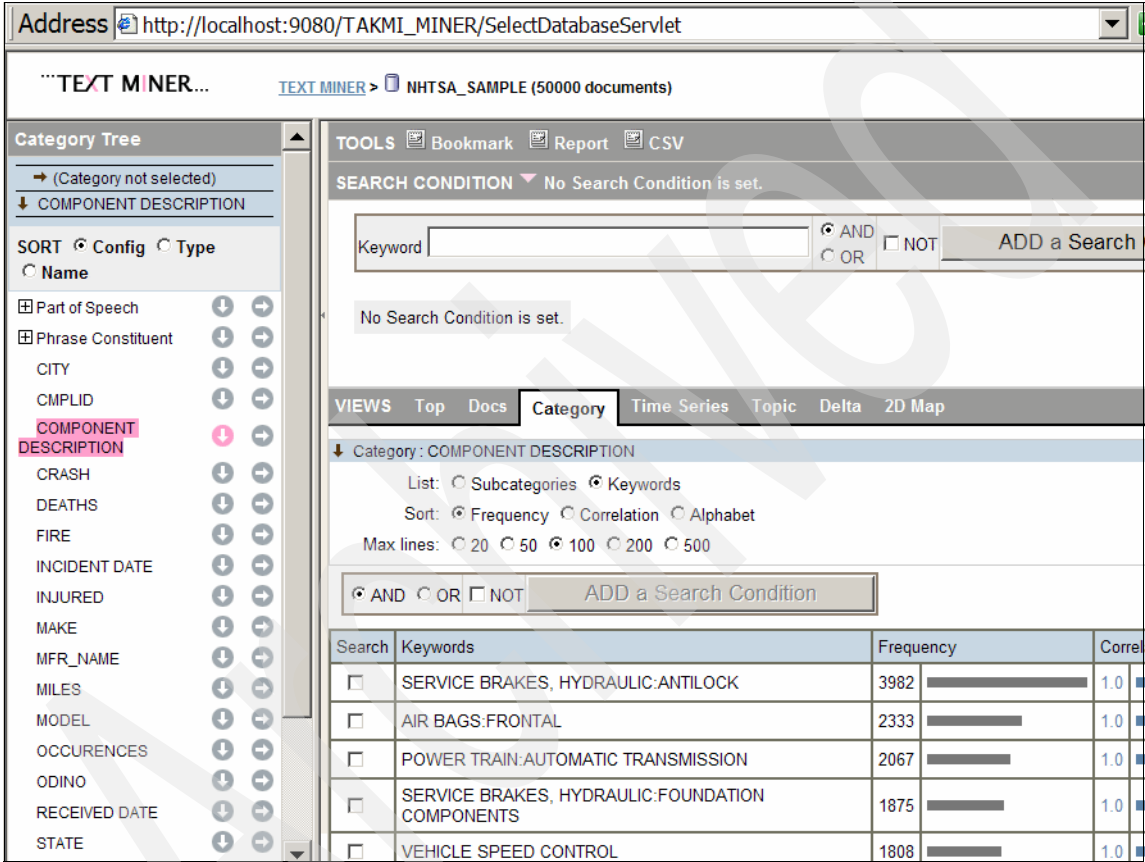


Figure 5-13 Category view with no search keyword condition

It is also possible to append an existing search condition with multiple keywords associated with a category. To explain this feature, in Figure 5-14, we begin with 94 documents identified for the keyword `fire`. In the Category view, we select `FUEL SYSTEM`, `GASOLINE` and `ELECTRICAL SYSTEM:IGNITION:SWITCH` as the keywords from the category called `COMPONENT DESCRIPTION`, apply the **AND** operator to update, and re-execute the new search condition.

Searched documents: 94 documents
 KEYWORD: fire 94 docs [DELETE](#)

VIEWS Top Docs **Category** Time Series Topic Delta 2D

Category: COMPONENT DESCRIPTION

List: ☐ Subcategories ☒ Keywords

Sort: ☒ Frequency ☐ Correlation ☐ Alphabet

Max lines: ☐ 20 ☐ 50 ☒ 100 ☐ 200 ☐ 500

☒ AND ☐ OR ☐ NOT [ADD a Search Condition](#)

Search	Keywords	Frequency
<input type="checkbox"/>	ENGINE AND ENGINE COOLING:ENGINE:GASOLINE	63
<input checked="" type="checkbox"/>	FUEL SYSTEM, GASOLINE	6
<input checked="" type="checkbox"/>	ELECTRICAL SYSTEM:IGNITION:SWITCH	6

Modified search results with multiple keywords

Searched documents: 12 documents

AND 12 docs [DELETE](#) KEYWORD: fire OR 1217 docs [DELETE](#) KEYWORD: FUEL SYSTEM, GASOLINE (COMPONENT DESCRIPTION) KEYWORD: ELECTRICAL SYSTEM:IGNITION:SWITCH (COMPONENT DESCRIPTION)

VIEWS Top Docs **Category** Time Series Topic Delta 2D Map

Category: COMPONENT DESCRIPTION

List: ☐ Subcategories ☒ Keywords

Sort: ☒ Frequency ☐ Correlation ☐ Alphabet

Max lines: ☐ 20 ☐ 50 ☒ 100 ☐ 200 ☐ 500

☒ AND ☐ OR ☐ NOT [ADD a Search Condition](#)

Search	Keywords	Frequency	Correlation
<input type="checkbox"/>	ELECTRICAL SYSTEM:IGNITION:SWITCH	6	8.3
<input type="checkbox"/>	FUEL SYSTEM, GASOLINE	6	20.9

Figure 5-14 AND operation on multiple category keywords

In Figure 5-14, we have now narrowed down the results to 12 documents in a database with 50000 documents where the keyword fire is mentioned with keywords called FUEL SYSTEM, GASOLINE and ELECTRICAL SYSTEM:IGNITION:SWITCH. This is very useful when you want to narrow down data to specific criteria before investigating individual documents in the result set for further research.

5.3.3 Date search

In date search, documents are identified by the a date criteria. The date criteria can be to list the documents created on a particular date or a list documents created over a specific date range.

You can search based on date in the Time Series, Topic, and Delta views of the Text Miner application. In Text Miner, the date criteria can be refined to Years, Months, Weeks, and days.

Figure 5-15 shows a Time Series view where the number of documents are listed per week starting from the 25th week of the year 1996 (1996 Year 25 Week) to the 34th week of 1996 (1996 Year 34 Week).

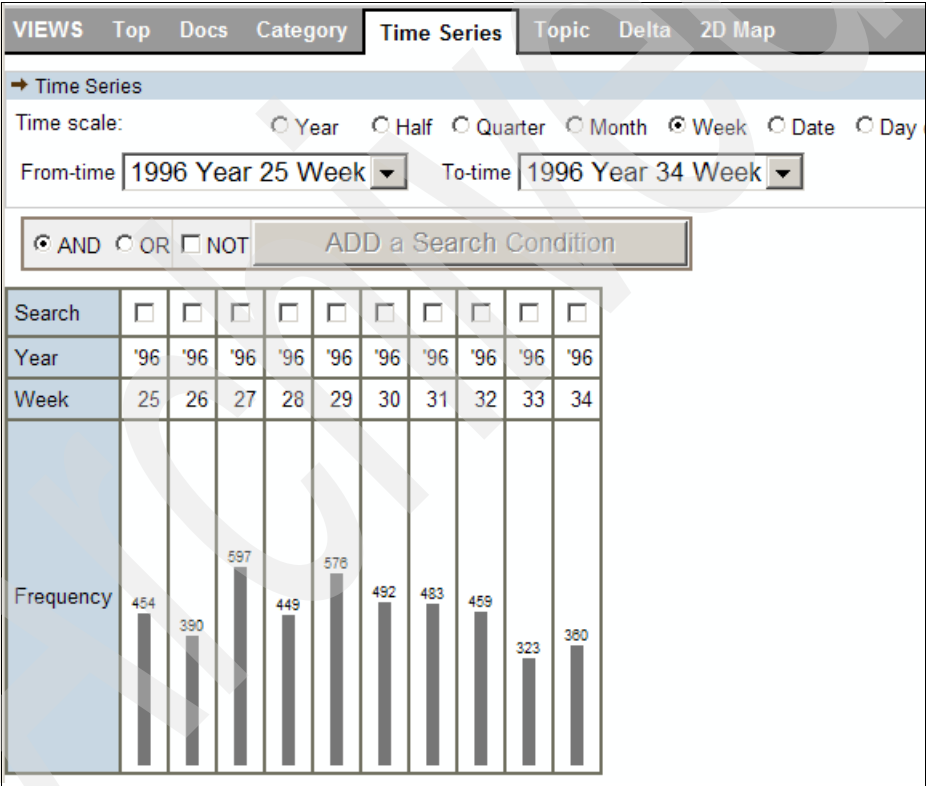


Figure 5-15 Sample time series search

This is very useful when you want to narrow the dataset to a particular time frame for further research and discovery.

5.3.4 Applying search operators

The user interface of Text Miner supports the combination of various search criteria (using keyword search, category search, and date search) to build a complicated search condition. The interface provides radio buttons to append AND, OR, and NOT search operators.

To illustrate the combination of various search criteria with operators, refer to Figure 5-16. We display a search condition where we narrow down complaints that mentioned the keywords accident AND injury, and the complaints occurred on Fridays OR Saturdays, OR Sundays. In other words, we want to examine all accident and injury related complaints that occur on a weekend, including Friday.

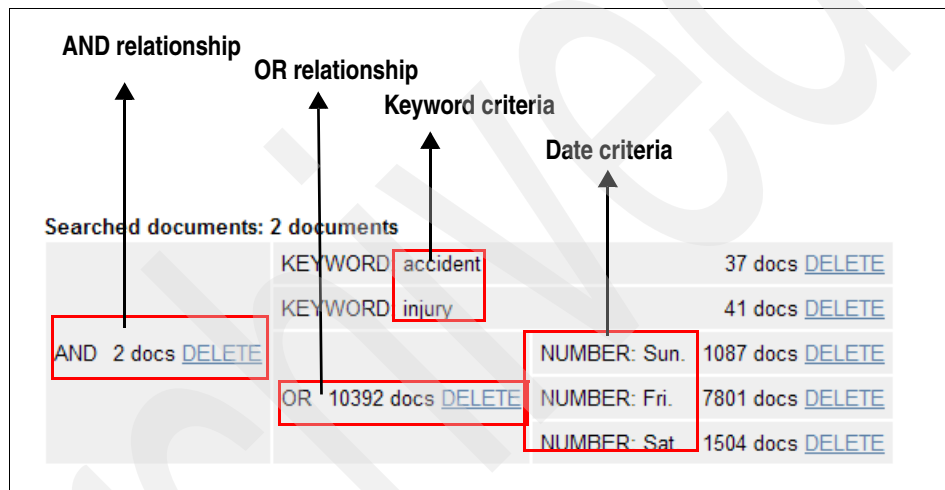


Figure 5-16 Overview of search

We describe how to use the Text Miner user interface to build complex search conditions with OR, AND, and NOT operators as follows:

- ▶ “Adding AND condition” on page 105
- ▶ “Adding OR condition” on page 106
- ▶ “Adding OR condition while an AND condition already exists” on page 107
- ▶ “Adding a search condition using multiple check boxes” on page 108
- ▶ “Adding a search condition from cells” on page 109

Adding AND condition

In Figure 5-17, we demonstrate applying an AND condition to an already existing search condition.

Searched documents: 94 documents
KEYWORD: fire 94 docs [DELETE](#)

1. Search condition before adding

Keyword

ignition

☒ AND
☐ OR
☐ NOT

ADD a Search Condition

2. Add keyword ignition as AND operation

Searched documents: 1 documents

AND 1 docs [DELETE](#)

KEYWORD: fire 94 docs [DELETE](#)
KEYWORD: ignition 59 docs [DELETE](#)

3. Condition after adding

Figure 5-17 Adding AND condition

First, we have an existing search condition that is based on the keyword fire. Second, we enter the keyword ignition, select the **AND** radio button, and click the **Add a Search Condition** button. The second step adds the AND condition to the existing search. Here is the result: We identified one document that refers to *ignition AND fire*.

Adding OR condition

In Figure 5-18, we demonstrate applying an OR condition to an already existing search condition.

Searched documents: 8 documents
KEYWORD: crash 8 docs [DELETE](#)

1. Search condition before adding

Keyword

collision

☐ AND
☒ OR
☐ NOT

ADD a Search Condition

2. Add keyword collision as OR operator

Searched documents: 19 documents

OR 19 docs [DELETE](#)

KEYWORD: crash 8 docs [DELETE](#)
KEYWORD: collision 11 docs [DELETE](#)

3. Condition after adding

Figure 5-18 Adding OR condition

First, we have an existing search condition based on crash. There are 8 documents related to crash. Second, we enter the keyword collision, select the **OR** condition, and click the **ADD a Search Condition** button. This adds an OR condition to the existing search. There are 11 documents that match the keyword collision. Here is the result of the OR condition: We obtained a total of 19 documents that refer to crash or collision. This feature is useful when we have not identified crash and collision as synonyms.

Adding OR condition while an AND condition already exists

In Figure 5-19, we demonstrate applying an OR condition to an already existing search condition that is built with an AND operator.

1. Condition before adding

Searched documents: 5 documents

AND 5 docs [DELETE](#)

KEYWORD: WIRING 509 docs [DELETE](#)

KEYWORD: fire 94 docs [DELETE](#)

2. Add keyword "ignition" as OR condition

Keyword ☒ AND ☐ OR ☐ NOT [ADD a Search Condition](#)

3. Condition after adding

Searched documents: 6 documents

AND 6 docs [DELETE](#)

KEYWORD: WIRING 509 docs [DELETE](#)

KEYWORD: fire 94 docs [DELETE](#)

OR 152 docs [DELETE](#)

KEYWORD: ignition 59 docs [DELETE](#)

Figure 5-19 Adding OR condition to existing AND condition

First, we have an existing search condition based on WIRING or fire using the AND condition. Then we enter a new keyword ignition to the existing condition, click the **AND** operator, and then click **ADD a Search Condition** button. This allows us to discover documents that contain the keyword WIRING and that are also mentioned with the keyword fire or ignition.

Adding OR condition to an existing query:

When an OR condition is added while an AND condition already exists, the last condition on the list that is subjected to the AND condition will be chosen to apply the newly appended OR condition.

In Figure 5-19 on page 107, WIRING and fire were in AND relation, resulting in documents where WIRING and fire occurred. Notice that the keyword fire is the last search criteria. When ignition is applied as an OR condition, the keyword fire in the initial condition is chosen for the OR operation. The resulting query returns documents where WIRING would occur with the occurrences of fire or ignition.

Adding a search condition using multiple check boxes

In Figure 5-20, we demonstrate how to apply multiple keywords from the Category and Time Series views to an existing search condition. We discover documents that refer to accidents and injuries that occurred on weekend days (friday, saturday, and sunday).

Selecting multiple check boxes:

When multiple check boxes in the Category view and Time Series view are selected to append to existing search conditions, the selected items are appended together with an OR operator. For multiple selections, OR is the default operator to extract a large set of documents.

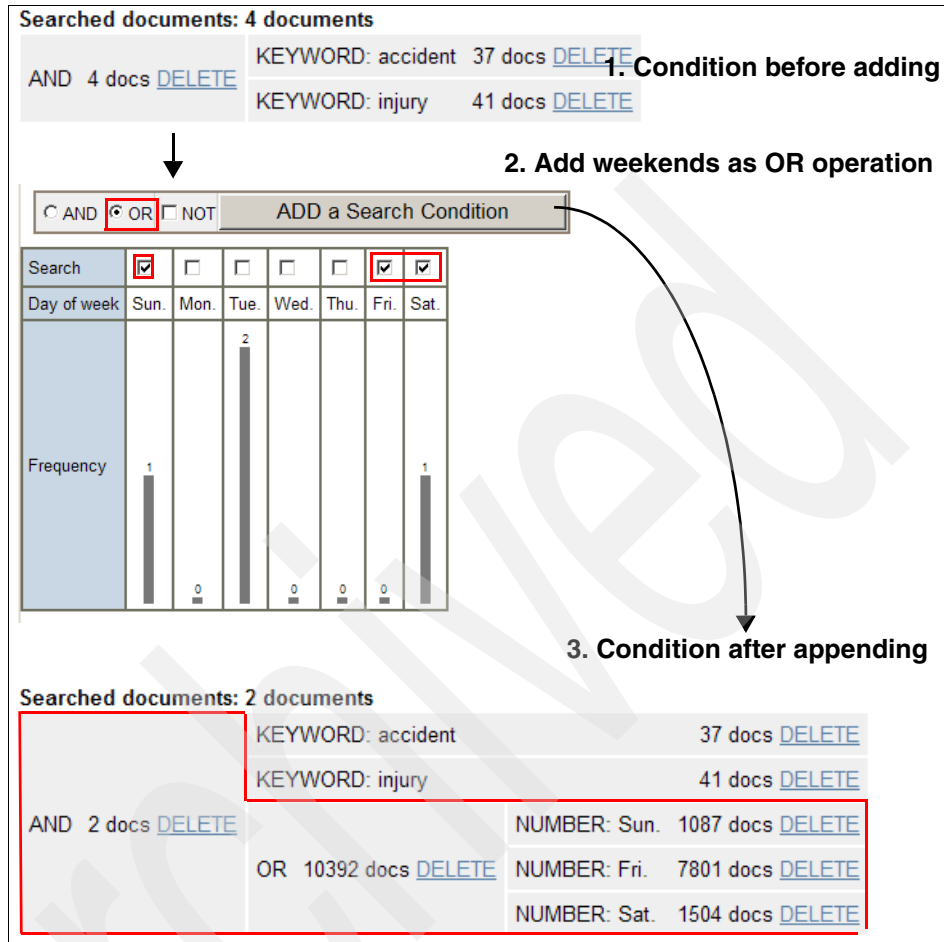


Figure 5-20 Adding search condition using multiple check boxes

First, we have an existing search condition based on the keywords accident AND injury. Second, we add weekends by selecting the **Sunday**, **Friday**, and **Saturday** check boxes, select the **OR** operator, and click **ADD a Search Condition** button. This gets us the result as shown in step number three of Figure 5-20: We discover two documents that refer to accident and injury and occurred over the weekend (Sunday, Friday, or Saturday). Notice that the Time Series view automatically adds an ADD to the existing condition.

Adding a search condition from cells

In Topic, Delta, and 2D Map views, the search results are displayed in a table. The values in the table represents the frequency of occurrences of documents with the keywords specified in the horizontal row and vertical column. The entries in the table are called *cells*. We explain these views later in this chapter.

In this section, we explain the creation of a search condition when a cell is clicked in Topic, Delta and 2D Map views.

In Figure 5-21, first, we have an existing condition based on the keyword crash. Second, we click a cell in the Topic view to discover documents that refer to crash related to STEERING:LINKAGES:TIE ROD ASSEMBLY in year 1995. By default, the clicking of the cell appends the original search condition with an AND operation on the fields related to the cell as shown in step number of three of the figure. The search button is not required to generate the new search condition.

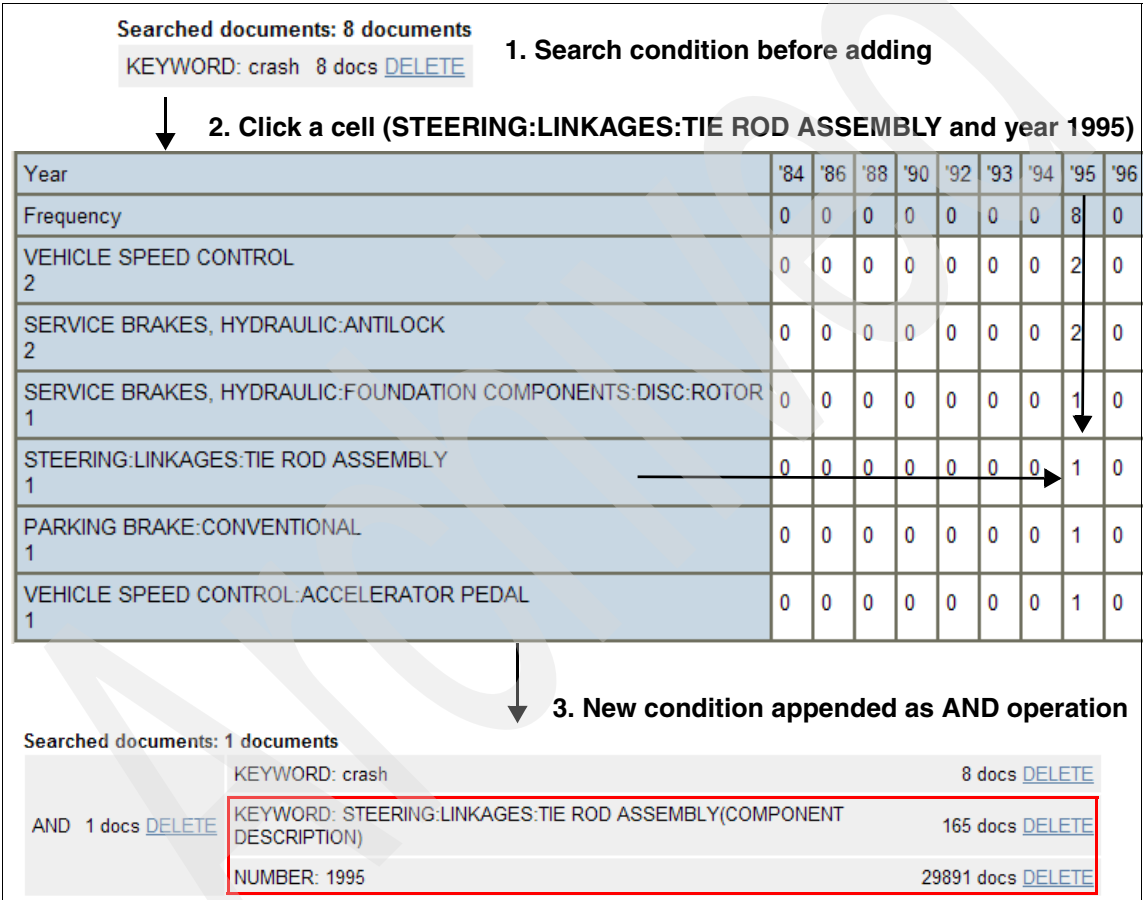


Figure 5-21 Add search condition by selection cells

This is very useful when you discover some interesting results in the cells and want to drill down and perform further research and discovery.

5.4 Views

In the earlier sections, while we explained the categories and search conditions, we used various views of Text Miner to display the results. This section is dedicated to explaining the views and their appropriate use.

Text Miner provides seven views:

- ▶ Top view
- ▶ Docs view
- ▶ Category view
- ▶ Time Series view
- ▶ Topic View
- ▶ Delta view
- ▶ 2D Map view

In this section, we explain each view, how it represents the results of a search condition in a unique way that aids in the process of discovering the business insights, anomalies, problems, or trends.

5.4.1 Top view

Figure 5-22 shows the Top view. This view is automatically displayed when you first entered the Text Miner application and selected a database. There is no mining function available in the Top view, but the view shows that the database is you selected is loaded and the OmniFind Analytics Edition index for the database is valid.

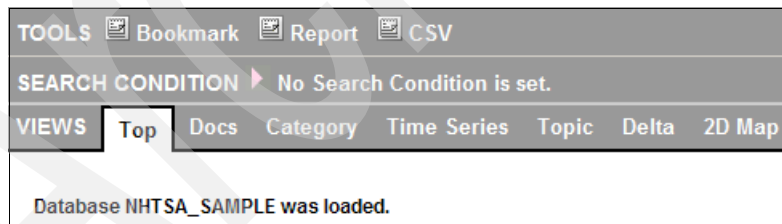


Figure 5-22 Top view

5.4.2 Docs view

In the Docs view, documents that meet the executed search conditions are listed in the order that they were created in the source database. The Docs view is used to access the source documents and discover the root cause of the identified condition.

In Figure 5-23, we display a sample Docs view with the documents that refer to the keyword fire.

Keyword

☒ AND
☐ OR

☐ NOT

ADD a Se

Searched documents: 94 documents
KEYWORD: fire 94 docs [DELETE](#)

VIEWS Top Docs Category Time Series Topic Delta 2D Map

↓ Documents
Docs per page: ☐ 10 ☒ 20 ☐ 50 ☐ 100

◀◀ 1 - 20 ▶▶

Page navigation links

Document display table

Title	1514
Text	DESCRIPTION: VEHICLE FIRE. *AK
Detail	DETAIL
Standard Date	19950913

Figure 5-23 Docs view with current search condition

The Doc view as shown in Figure 5-23 on page 112 has the following functions and features:

- ▶ Docs per page (radio buttons): You can select one of the radio buttons to indicate the number of documents to be displayed per page.
- ▶ Page navigation links: You can click the arrow links to navigate through the pages of the result set. Click the double-left-arrow link to go back to the first page of the result set. Click the double-right-arrow link to go to the last page of the result set. Single-left arrow link enables you to go back one page from the current result set and right-arrow link enables you to go forward a page in the result set. Documents are listed in the order they appear in the source data.
- ▶ Document display table (shown as Document Inspector). This includes summary information about a particular document. It usually comes with the following fields displayed:
 - Title: The row labelled `Title` displays the title of the document. The title is specified when importing the data. Refer to 4.5, “Generating ATML files” on page 70 to specify the data field that will represent the value of title.
 - Text: The text of the document that is displayed in the row labelled `Text`. When you import CSV documents to ATML, you specify the field that will be used as `Text`. For our NHSTA sample data, we used the `DESCRIPTION` field as the `Text` to be displayed here.
 - Detail: The link labelled `Detail` launches a new window called Document Inspector which shows the detailed information about the document. Figure 5-24 shows a sample window from the Document Inspector. The Document Inspector window shows the document’s ID, Title, Text, standard category information, and keywords extracted by natural language processing. By displaying the document in the Document Inspector, you can inspect the particular document. It also help you to understand better the keywords, their association to the system category names that were specified during the data ingestion and the categories assigned by natural language processors.
 - Standard Date: Every document has a data associate to it. This date is used for Time Series, trend analysis. You specify which date field in the source data to be the `Date` field during data ingestion.

Note: In order to comply with privacy to individual and company-specific data, we have darkened out a few critical fields in Figure 5-24.

Document Inspector

Document Information	
Document ID	1514
Title	1514
Text	DESCRIPTION: VEHICLE FIRE. *AK

Standard Information	
Category	Keyword
CITY	PLEASANT H
CMPLID	1514
COMPONENT DESCRIPTION	ENGINE AND ENGINE COOLING:ENGINE:GASOLINE
CRASH	
DEATHS	0
FIRE	Y
INCIDENT DATE	19950220
INJURED	0
MAKE	ITSO TURBO SERIES
MFR_NAME	ITSO CAR COMPANY
MILES	
MODEL	MODEL A
OCCURENCES	3
ODINO	481611
STATE	NC
VIN	1122334455
YEAR	1990
Standard Date	19950913

Keyword Information				
Category	Keyword	Highlight	Beginning Offset	Ending Offset
Common Noun	vehicle		0	7
Common Noun	fire		8	12
Proper Noun	Alaska		16	18

Figure 5-24 Document Inspector sample

Next we describe the entries on the Document Inspector window displayed in Figure 5-24:

- ▶ **Document Information:** The Document Information area shows the document ID, title, and the text in the source data that is imported by OmniFind Analytics Edition.
- ▶ **Standard Information:** This area contains fields that are associated with the document when it is imported, as specified in <category, keyword> pairs. When there are multiple keywords for a standard item category, multiple lines are used to show all these keywords. Refer to 4.4, “Database category creation” on page 63 where the standard item categories are defined in category_tree.xml and database_config.xml files.
- ▶ **Keyword Information:** In the Keyword Information area, keywords extracted by the language processing are displayed as <category, keyword> pairs. Clicking a radio button in the highlight field will highlight the corresponding keyword in the original text.

Highlighting the document with search keywords:

- ▶ When keywords or character strings are specified as search conditions, parts of the document that meet the search terms are highlighted.
- ▶ When phrases are specified as search conditions, the entire phrase will be highlighted.

5.4.3 Category view

The Category view in Text Miner is used to list the search results by the keywords associated to a specified category. The view displays the frequency and correlation of the keywords associated to a search condition and the selected category or subcategory.

Note: You should always be looking for keywords with high correlation or high frequency for in-depth analysis. *Correlation* indicates the association between two keywords. A correlation value that is higher than 1.0 represents anomalies and usually require further investigation. It is important for the analyst to pay attention to any entries with a high correlation value.

A category or a subcategory from the category tree has to be selected to use this view. In Figure 5-25 we display the Category view for search results on the keyword `fire` and the documents associated with the `COMPONENT DESCRIPTION` category.

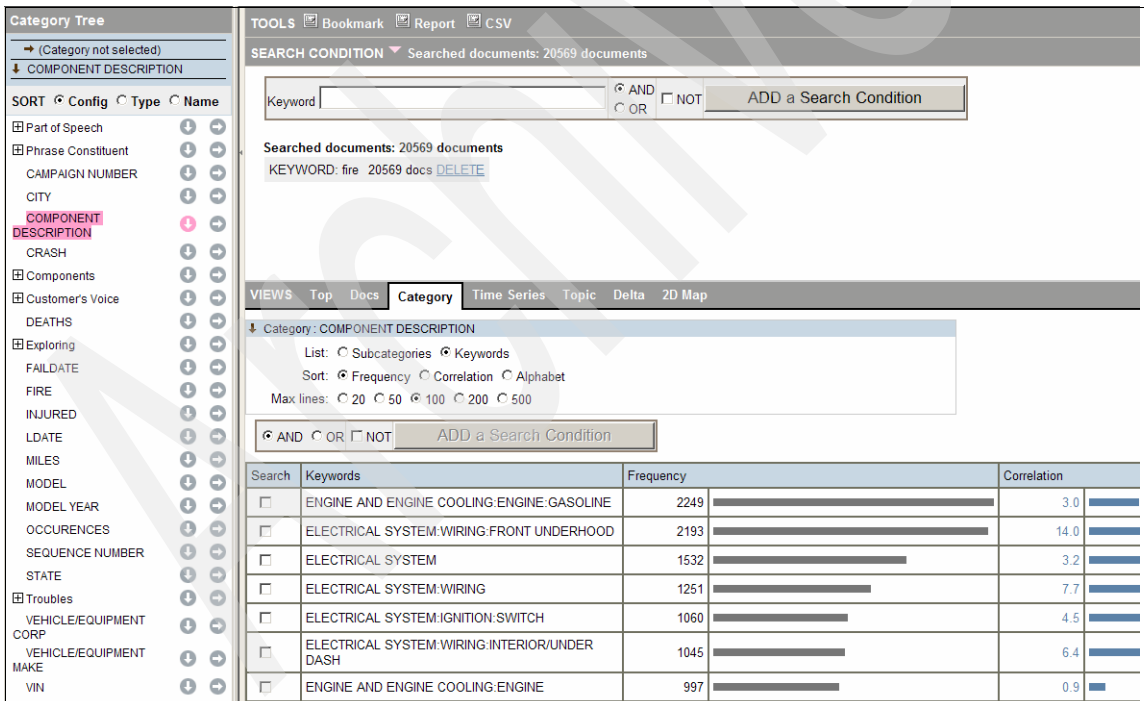


Figure 5-25 Category view

Understanding the results in the Category view:

In Figure 5-25, we search against 20569 documents for the keyword `fire` and the documents that are associated with the `COMPONENT DESCRIPTION` component.

When the results are listed by frequency, it shows that there are 2249 documents out of the identified 20569 documents contain the keyword called `ENGINE AND ENGINEER COOLING:ENGINE:GASOLINE`. Furthermore, the Category view shows that the correlation value associated with that entry is 3.0. In other words, there is a high correlation between `ENGINE AND ENGINEER COOLING:ENGINE:GASOLINE` and `fire`. Remember that a correlation value that is higher than 1.0 represents anomalies. In other words, there is a high anomalies between `ENGINE AND ENGINEER COOLING:ENGINE:GASOLINE` and `fire`.

In addition, there is a high frequency, 2249 of documents that are associated with `fire` and `ENGINE AND ENGINEER COOLING:ENGINE:GASOLINE`. In this particular case, you probably can guess that it is very likely that the fire is caused by gasoline and gasoline pumps and the engine. So our finding regarding a high correlation between `fire` and gasoline or engine might not provide any new insights. We can, however, drill down the search to see what type of car makes, models, or during what period that there are high frequencies and high correlations between `GASOLINE` and `fire`.

Some other findings from the sample Category view might be more surprising. Looking at the second entry on the Category view, we discover a *high* correlation value of 14.0 between the keyword `fire` and keywords `ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD`. Fire incidents due to wiring are not as obvious as gasoline. In this case, you would want to drill down on this entry to identify the root cause of the high correlation (by selecting the check box next to the second entry; it adds this entry to the existing search).

For more information about correlation, refer to 6.3, “Correlation” on page 145.

With reference to Figure 5-25, the following list describes the functions available on the category view:

- *List* radio buttons: Use the List radio buttons to display the results by Subcategories (of the selected category), or Keywords.

In Figure 5-26, we display the Category view for the keyword `fire` within the category called `Part Of Speech`. The figure shows the results sorted by keywords.

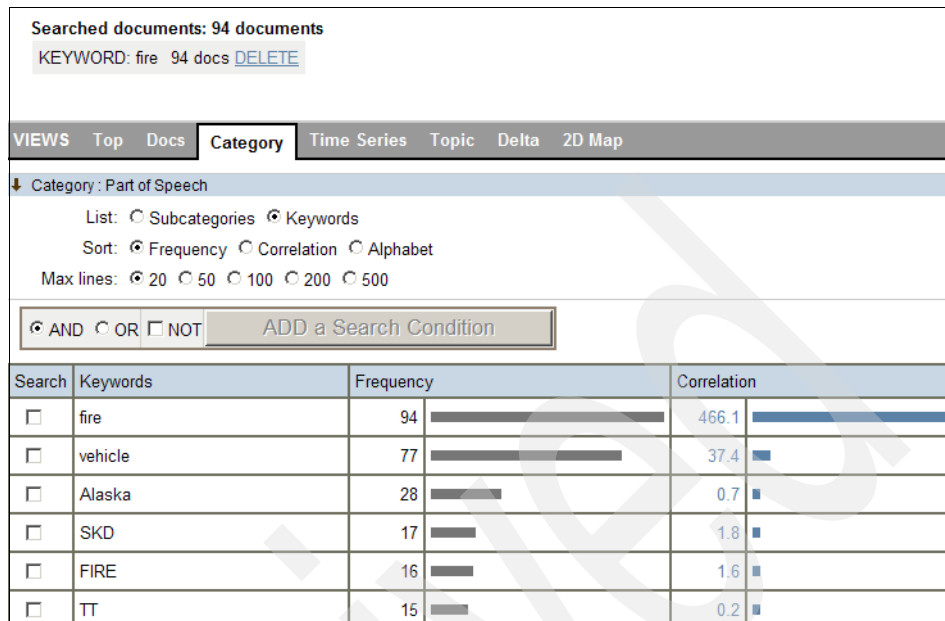


Figure 5-26 Display results sorted by keywords

The category called Part of Speech (POS) has subcategories called Noun, Verb, Adjective, and Conjunction. In Figure 5-27, we display the same set of results sorted by the subcategories.

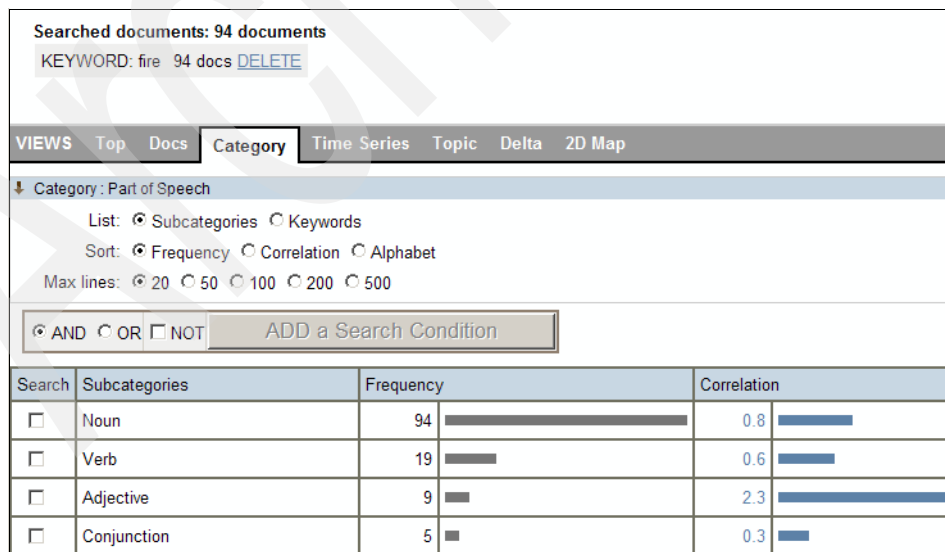


Figure 5-27 Display results sorted by Subcategories

- ▶ *Sort* radio buttons: Use the Sort radio buttons to list the keywords by Frequency, Correlation, or Alphabets:
 - Frequency: The keywords and subcategories are listed in the order of how often they occur in the result set. The higher the frequency, the higher the subcategory is on the list. Figure 5-27 is sorted by frequency.
 - Correlation: The result set is listed in the order of correlation strengths with the specified keywords or subcategories. This is very useful to identify any anomalies between the keywords and subcategories, especially when you are not sure what you are looking for in the dataset and want to discover any anomalies that are unexpected.
 - Alphabet: The keywords and subcategories are sorted alphabetically.
- ▶ *Max lines* radio buttons: Use the radio buttons to specify the maximum number of keywords or subcategories to be listed.
- ▶ *Search check box* (inside the table). The search condition can be appended by selecting multiple keywords listed in the Category view. You can select multiple check boxes and click the **Add a search condition** button at the top of the window to append or create a new search condition. If a search condition already exists, the selected keywords are appended with the OR operator. Refer to Figure 5-14 on page 103 where we show the flow of instructions for selecting multiple keywords and the appended search condition. This is useful if you want to further investigate the result set that are associated with the particular keywords.
- ▶ *Frequency columns* in the results table: The column labelled Frequency in the results table displays the count of occurrences of the keywords as a number and bar graph. This is useful to quickly identify how often a particular keyword or keywords appear in the result set.
- ▶ *Correlation* (inside the table): The columns labelled Correlation display the strength of correlation between the set of result sets (retrieved documents) and the keywords. The correlation count is represented as a number and bar graph. If the number is higher than 1.0, it deserves special attention for further investigation.

5.4.4 Time series view

The Time Series view displays the frequency count of the search results over a period of time. The Time series view can be used by analysts to understand the occurrences of particular keywords associated with time.

For example, in Figure 5-28 we display the occurrences of 94 fire related incidents isolated by the year. We discover an abnormal increase in the fire related incidents in year 1995 (increase from 10 in year 1994 to 50 in 1995) and

a conservative decline in year 1996 (from 50 in year 1995 to 34 in year 1996). This view is important to understand the impact of a change on a time scale.

For further in-depth analysis, the year 1995 can be added to the search condition to access the reported incidents.

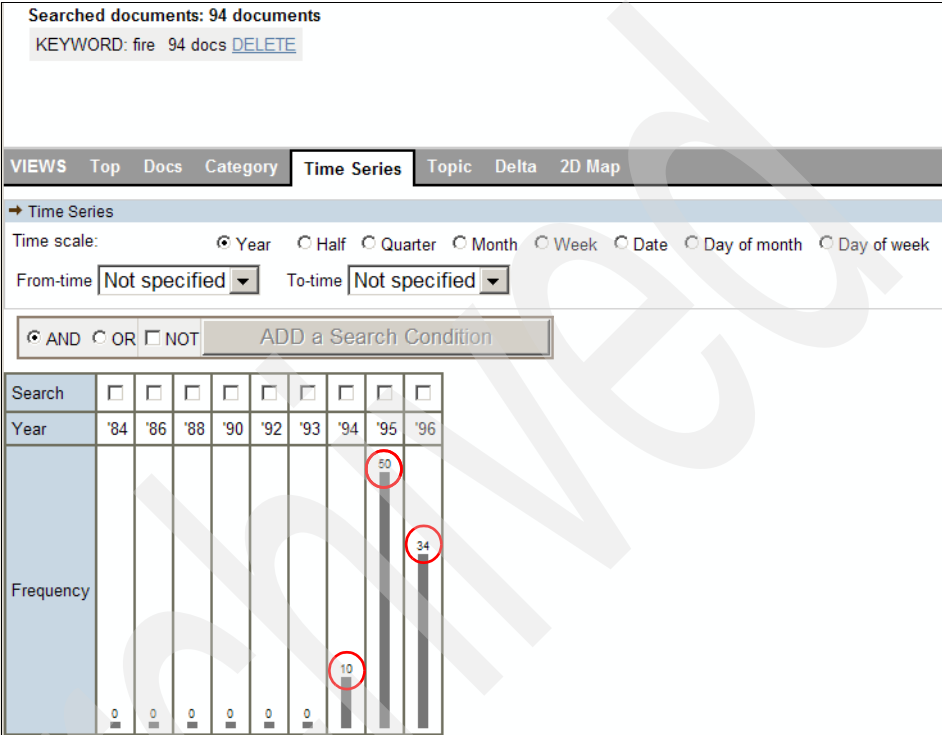


Figure 5-28 Time series view

Understanding the results on the Time Series view:

In Figure 5-28, we display the 94 complaints that contain the keyword fire and occurred between the years 1984 and 1996. The positive frequency counts are highlighted in Figure 5-28. This view is ideal for analysts to identify abnormal trends or study the impact of some positive fix. A sudden increase in the frequency during a particular period of time usually requires further investigation. This might be where you can discover that a bad auto component produced in a certain year is causing a lot of problems.

With reference to Figure 5-28 on page 120, the following list describes the functions available in the Time series view:

- ▶ *Time scale* radio buttons: The Time scale radio buttons helps to refine the time period by Year, Half, Quarter, Month, Week, and Date. The terms Half and Quarter mean half of the year and quarterly.
- ▶ *From-time* drop-down list box: The drop-down list box enables you to specify from which time period to start the display range.
- ▶ *To-time* drop-down list box: The drop-down list box enables you to specify to which time period to end the display range; for example, when you identified some anomalies in 1995. You can select the From-time and To-time range to be within 1995 and set the Time scale to Month to further narrow down and see in which months most incidents occur.
- ▶ *Search check box* in the results table: The search condition can be appended or created by selecting Multiple Search check boxes. You can select multiple check boxes and click the **Add a search condition** button to append or create a new search condition. If a search condition already exists, then the selected dates will be appended with an OR operator. Refer to Figure 5-20 on page 109 where we show the flow of instructions for selecting multiple items and the appending to search condition.

5.4.5 Topic view

In the Topic view, the frequency counts and deviations of the keywords are displayed over a period of time. Unexpected spikes in the deviations are marked by pink bars. The highlighted spikes are ideal alerts for the analysts to do an in-depth analysis.

When the Topic view is displayed, average occurrence of the keyword on the Time scale is computed across the data in the dataset. Any frequency count that deviates (increases or decreases) from the computed average are highlighted in pink. Refer to 6.4, “Topicality index” on page 152 for more details on the computation.

In the Topic view shown in Figure 5-29, we display keywords for a category called COMPONENT DESCRIPTION. The search condition in the sample refers to 20569 incidents that have occurrences of the keyword fire. The sample database has 501376 documents in total and the data is collected from 1984 to 2007 (14 years).

With that said, notice the highlighted circles where cells with the frequency counts of 675 and 90 are marked. The highlighted circles alert the analysts that the frequency count of fire related incidents that occurred in the years 1996 and 1997 are above the average frequency counts when computed over the 14 years of data. The analysts can select any of the highlighted circles to narrow down the documents and discover the root cause.

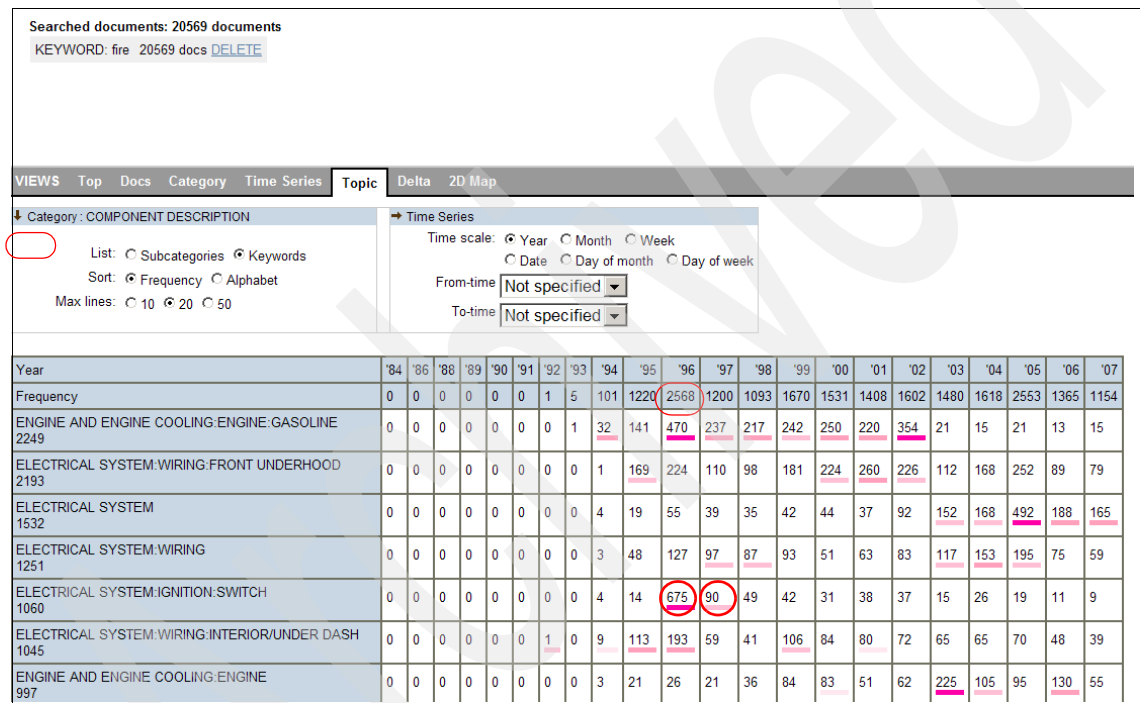


Figure 5-29 Topic view

Understanding the results in the Topic view:

In the Topic view shown in Figure 5-29, we display keywords for the COMPONENT DESCRIPTION category over the 20569 incidents that contain the keyword fire. The result is listed by the frequency of the keywords appeared and is sorted by the year.

The following list describes the significance of each number shown in Figure 5-29:

- ▶ *Year row*: This row (the first row) marked the year number (84 to 07) of the data.
- ▶ *Frequency row*: The row (the second row) shows the occurrences of the keyword for a particular year. For example, the number 2568 that is highlighted indicates that 2568 incidents of fire were registered in the year 1996 dataset.
- ▶ *Category keywords and frequencies*: Right below the cell marked Frequency, the keywords for the selected category (COMPONENT DESCRIPTION) are listed by how often they occur in the dataset. The numbers below each keyword indicate the frequency counts. For example, the number 1060 that is highlighted for keyword ELECTRICAL SYSTEM: IGNITION: SWITCH indicates that there are 1060 documents that contain the keyword ELECTRICAL SYSTEM: IGNITION: SWITCH and fire.
- ▶ *Cells in the table*: The cells in the table refer to the frequency counts across the year on the vertical scale and the keyword on the horizontal scale. For example, the number 675 highlighted for the intersection of year 1996 and keyword ELECTRICAL SYSTEM: IGNITION: SWITCH indicates that there are 675 incidents were reported with the keywords fire and ELECTRICAL SYSTEM: IGNITION: SWITCH.
- ▶ *Cells highlighted with pink bars*: The unexpected deviations in the frequency counts are highlighted by pink bars. The bars are intended to catch the attention of the analysts. This is where you want to drill down to do further analysis.

With reference to Figure 5-29, the following list describes the functions available on the Topic view:

- ▶ *List radio buttons*: Use the List radio buttons to display the results by Subcategories (within the selected category) or Keywords.

In Figure 5-29, we display the Topic view for the keyword fire and the category called COMPONENT DESCRIPTION. The figure displays the results sorted by keywords.

- ▶ *Sort* radio buttons: Use the Sort radio buttons to list the keywords by Frequency, Correlation, or Alphabets:
 - Frequency: The keywords and subcategories are listed in the order of how often they occur in the result set. The higher the frequency, the higher the subcategory is on the list. Figure 5-27 is sorted by frequency.
 - Correlation: The result set is listed in the order of correlation strengths with the specified keywords or subcategories. This is very useful to identify any anomalies between the keywords and subcategories, especially when you are not sure what you are looking for in the dataset and want to discover any anomalies that are unexpected.
 - Alphabet: The keywords and subcategories are sorted alphabetically.
- ▶ *Max lines* radio buttons: Use the radio buttons to specify the maximum number of keywords or subcategories to be listed.
- ▶ *Time scale* radio buttons: The Time scale radio buttons helps to refine the time period by Year, Half, Quarter, Month, Week, and Date. The terms Half and Quarter mean half of the year and quarterly.
- ▶ *From-time* drop-down list box: The drop-down list box enables you to specify from which time period to start the display range.
- ▶ *To-time* drop-down list box: The drop-down list box enables you to specify to which time period to end the display range. For example, instead of displaying data over a 10 year period, you can set the From-time and To-time to show data from the last 5 years only.
- ▶ *Search check box* in the results table: The search condition can be appended or created by selecting Multiple Search check boxes. You can select multiple check boxes and click the **Add a search condition** button to append or create a new search condition. If a search condition already exists, then the selected dates will be appended with an OR operator. Refer to Figure 5-20 on page 109, where we show the flow of instructions for selecting multiple items and the appending to search condition.
- ▶ *Selectable cells* of the results table: Each cell in the results table is clickable to generate a new search condition. The new search condition results in appending the horizontal and vertical fields associated to the selected cell to the existing search condition with AND operator. If no search condition exists, then a new search condition is created.

In Figure 5-30, we show the Topic view with keywords from 675 documents that results by clicking the cell (refer to Figure 5-29) that is an intersection of year 1996 and the keyword called ELECTRICAL SYSTEM:IGNITION:SWITCH. For more information about search condition by selecting cells, refer to 5.3.4, “Applying search operators” on page 105.

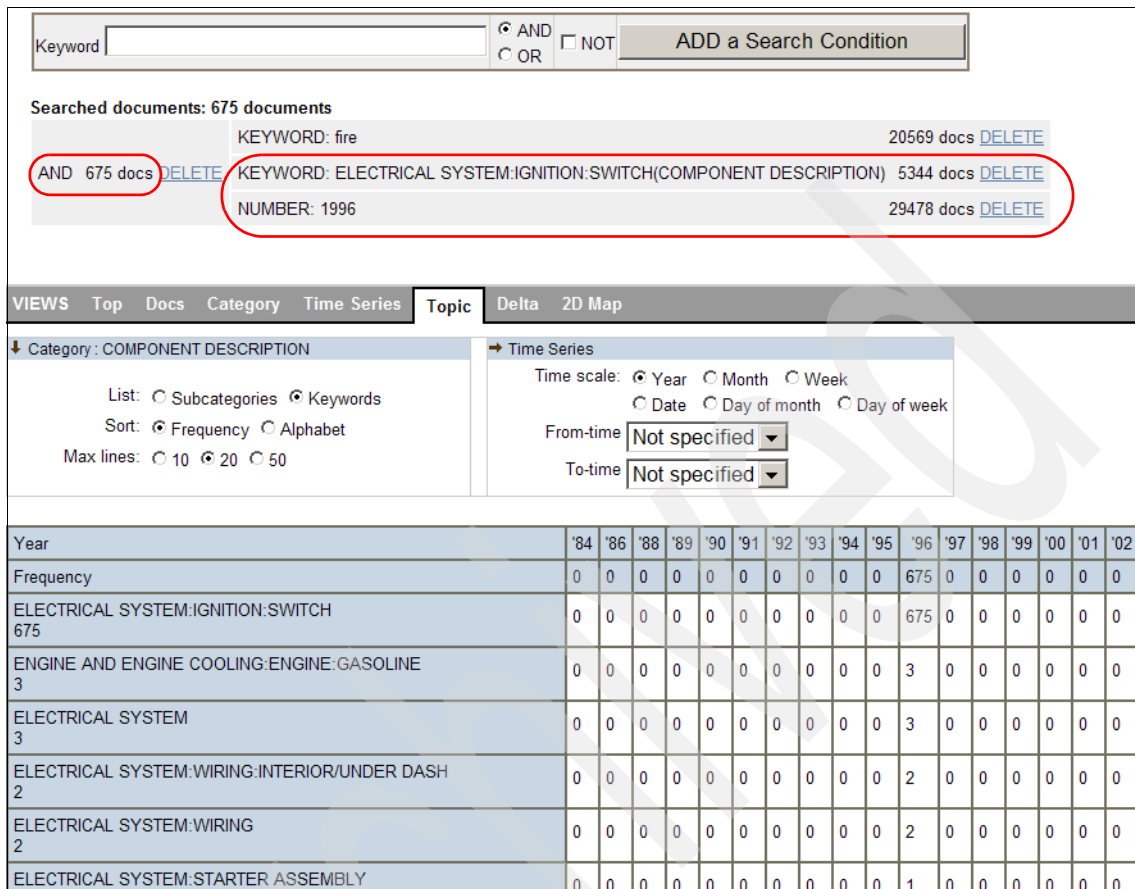


Figure 5-30 Results of cell selection

5.4.6 Delta view

In the Delta view, the keyword frequencies and increase indicators are displayed against a time scale. The increase indicators refer to the increase in the frequencies of keywords and subcategories along a selected category. This is achieved by measuring how much the frequency on each given date deviates from a constant frequency.

The increase indicators help identify an increasing or decreasing trends in future frequencies of the keywords and subcategories. Keywords with abnormal increase of frequency are marked in pink to alert analysts about possible anomalies. Refer to 6.5.1, “Increase indicator index in the Delta view” on page 159 and 9.3.2, “Increase Detection results” on page 236 to learn more about understanding and configuring alerts for unexpected increase indicators.

In Figure 5-31, we show the Delta view of keywords associated to the COMPONENT DESCRIPTION category found across the 20569 documents that contain the keyword fire.

Notice the increased indicators highlighted in circles for the column called ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD. The highlighted numbers 46.4 indicate a high increase in the usage of the keyword ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD from the years 1995 and 2000 when compared to the frequency counts in the recent past years.

The high number of 46.4 alerts the analysts that a problem has been detected with ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD in year 1995 and the root cause might be associated with the fire incidents. If the root cause of the problem is not addressed, the incidents of fire and ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD in the following years will increase, as we have seen. This increase is an indication to analysts as an early warning of anomaly. Further investigation should be done to drill down to the root cause of the problem.

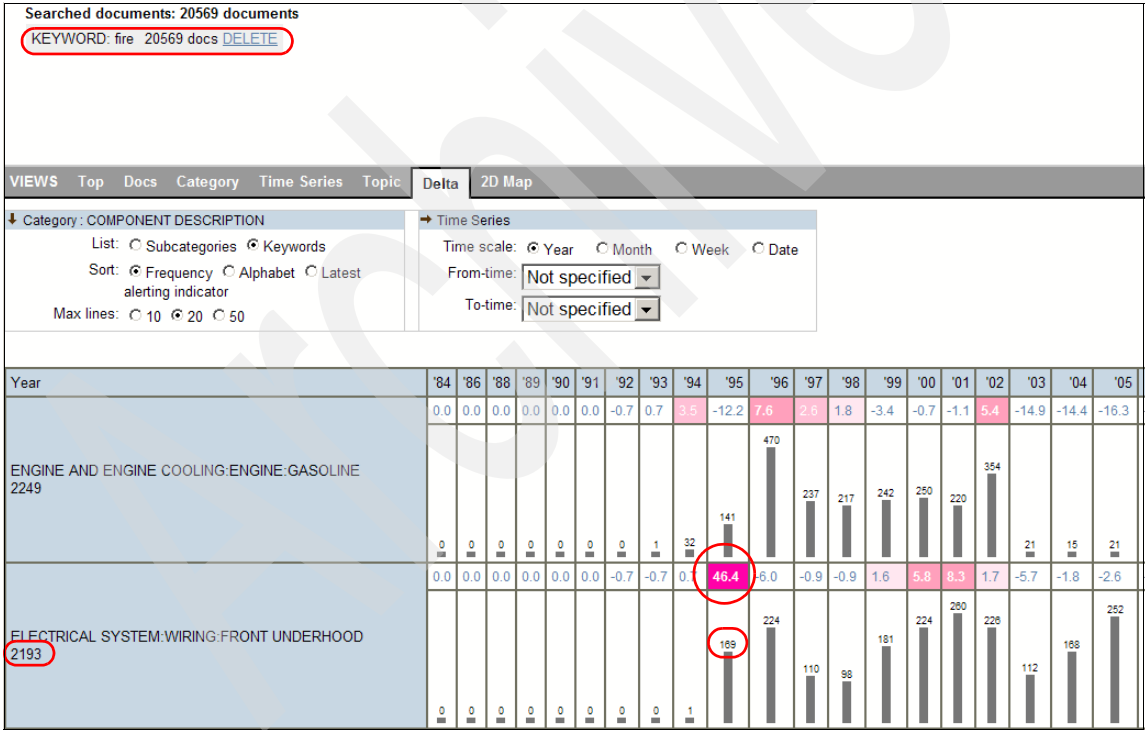


Figure 5-31 Delta view

The Delta view uses the same computation to determine the increase indicators as the Alerting System provided in OmniFind Analytics Edition. TAKMI Alerting System is a J2EE Web application in OmniFind Analytics Edition to monitor increases in the frequency indicators and correlation between keywords over a specified time period. Refer to Chapter 9, “Alerting System” on page 221 to learn about configuring the alerts.

It is important for analysts to see the anomalies indicated by OmniFind Analytics Edition through the higher number of keyword usage and the pink colored cells. Further research should be done with this information to prevent future problems.

Understanding the results in the Delta view:

In the Delta view shown in Figure 5-31, we display keywords for the COMPONENT DESCRIPTION category across the 20569 incidents that also contain the keyword fire. The results are sorted by the frequency of the keywords and are listed by the year.

The following list describes the significance of each number shown in Figure 5-31:

- ▶ *Year* row: This row (the first row) marked the year number (84 to 07) of the data.
- ▶ *Keyword* rows: Each row in the table represents the frequency count (by the bar graph) and the increased indicators for keywords found for the selected category/subcategory. Also the table cell with the keyword displays the frequency count. For example, in Figure 5-31, the number 2193 for the keyword ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD indicates that 2193 out of 20569 documents in the dataset contain this keyword are associated with fire.
- ▶ *Cells* in the table: The cells in the table display the frequency counts and increase indicators across the time unit on the vertical scale and the keyword on the horizontal scale. For example, in Figure 5-31, consider the cell at the intersection of the year 1995 and keyword ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD. The numbers 169 and 46.4 represent the frequency count and increased indicators for the keyword ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD in the year 1995.
- ▶ *Cells highlighted with pink shades*: The unexpected increase in the indicators are highlighted by pink colors. The varying shades of the pink color on various cells are intended to catch the attention of the analysts.

With reference to Figure 5-31, the following list describes the functions available on the Delta view:

- ▶ *List* radio buttons. Use the List radio buttons to display the results by Subcategories or Keywords.

In Figure 5-31, we display the view for the search keyword *fire* for the COMPONENT DESCRIPTION category. The results are listed by keywords.
- ▶ *Sort* radio buttons. Use the Sort radio buttons to sort keywords by Frequency, Alphabets, or Latest alert indicators:
 - Frequency: When selected, keywords and subcategories are listed in the order of how frequently they appear in the result set.
 - Alphabet: When selected, keywords and subcategories are sorted alphabetically.
 - Latest alerting indicator: When selected, keywords and subcategories are listed in the order of most recent occurrence of the increase in indicator.
- ▶ *Max lines* radio button. Use the Max line radio buttons to specify the maximum number of keywords or subcategories to be listed per page.
- ▶ *Time scale* radio buttons. The Time scale radio buttons helps to refine the time period by Year, Half, Quarter, Month, Week, and Date. The terms Half and Quarter mean half of the year and quarterly.
- ▶ *From-time* drop-down list box. The drop-down list box enables you to specify from which time period to start the display range.
- ▶ *To-time* drop-down list box. The drop-down list box enables you to specify to which time period to end the display range.

5.4.7 2D Map view

The 2D Map view uses a two-dimensional table to show the correlation between keywords that belong to a pair of categories (or subcategories). Table entries displaying a high correlation between the selected categories (or subcategories) are highlighted in pink. The highlighted pink table cells are the anomalies that an analyst should select for in-depth analysis.

For example, in Figure 5-32, we show the selection of category called COMPONENT DESCRIPTION (vertical arrow) and subcategory Noun Sequence (horizontal arrow) to generate the 2D map as shown in Figure 5-33.

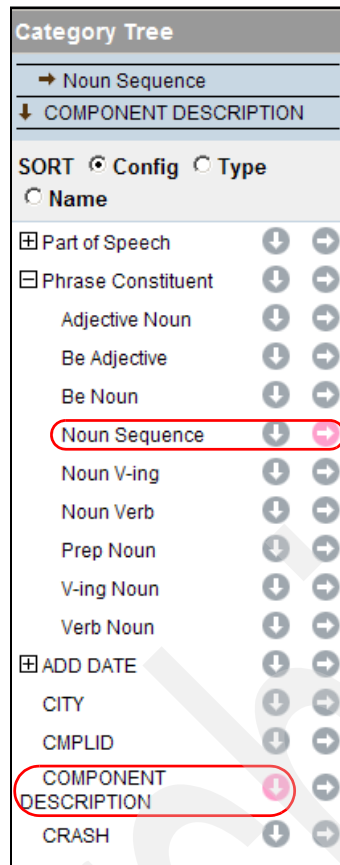


Figure 5-32 Selection of multiple categories for 2D map view

In Figure 5-33, we show the 2D Map that contains a table with keywords from the selected categories. There are quite a few table cells that are highlighted in different shades of pink color. As an analyst, one should identify large correlations in unexpected scenarios.

For example, in Figure 5-33, a correlation of 16.8 at the intersection of the keyword AIR BAGS:FRONTAL and BAG DID NOT DEPLOY is quite a concern for passenger safety. An analyst should select this cell to identify the documented incidents and narrow down the root cause of the failure. During the in-depth analysis, the analyst can also figure out the car make and models that were involved in the report.

The correlation numbers on the 2D Map view are generated by factoring a pre-configured set of top 100 keywords along the horizontal and vertical axes, but the Alerting System Web application is much more robust and handles up to 1,000 keywords x 12,000 keywords to be used for computing the correlation. The Alerting System in OmniFind Analytics Edition provides more details about configuring the variables to compute for correlations.

Understanding the results in the 2D Map view:

In the 2D map view shown in Figure 5-33, we display a two-dimensional table view of keywords for a category called COMPONENT DESCRIPTION (vertical axis) and a subcategory called Noun Sequence (horizontal axis).

The following list describes the significance of each number shown in Figure 5-33:

- ▶ *Row and column headers.* The headers for each row and columns represent the keywords from the selected categories.
- ▶ *Numbers in row and column headers:* The numerical numbers in the row and column headers represent the frequencies of the keywords in the entire dataset. For example, the highlighted numbers 2333 and 682 represent that there are 2333 documents in the database that contain the category keyword called AIR BAGS:FRONTAL, and 682 documents in the database contain the keyword called BAG DID NOT DEPLOY.
- ▶ *Cells in the table.* The cells in the table display the frequency counts and computed correlation numbers for the keywords on the horizontal and vertical axes. For example, in Figure 5-33, consider the cell at the intersection of the keywords AIR BAGS:FRONTAL and BAG DID NOT DEPLOY. The numbers 584 and 16.8 represent the frequency count and correlation numbers.
- ▶ *Cells highlighted with pink shades.* The cells with unexpected correlation numbers are highlighted in varying shades of pink colors. The varying shades of the pink color on cells are intended to catch the attention of the analysts. The cell with the strongest pink shade is the most important anomaly to investigate.

With reference to Figure 5-33, the following list describes the functions available on the 2D Map view:

- ▶ *List radio buttons.* Use the list radio buttons to display the results by Subcategories of the selected category or the keywords.

- ▶ *Sort* radio buttons. Use the Sort radio buttons to sort keywords and categories in the list by frequency or alphabetic order:
 - Frequency: When selected, keywords and subcategories are listed in the order of how frequently they appear in the result set.
 - Alphabet: When selected, keywords and subcategories are sorted alphabetically.
- ▶ *Max lines* radio buttons. Use the Max lines radio buttons to specify the maximum number of keywords or subcategories to be listed per page.

5.5 Reporting features

In this section, we explain the Text Miner features that can help store and restore search results discovered in various navigational views. These reporting features can be exploited to share the search findings with the effected parties. In Text Miner, you can bookmark a result set, generate a report for the result set, or export the result set to a CSV formatted file. The generated CSV file can be read with Microsoft Excel application.

Figure 5-34 shows the features provided in the Tools view.

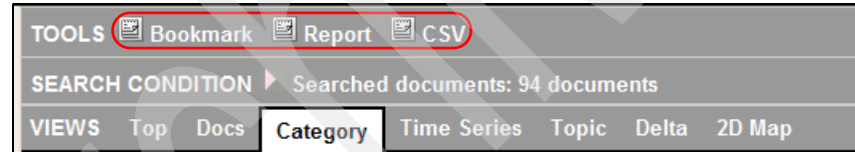


Figure 5-34 Tools view

5.5.1 Bookmark results

The bookmark function saves the current search conditions and parameters in the currently displayed view as a local HTML file. A connection to the server is reestablished when the saved bookmark is opened for viewing and hence allows the analysis to resume with the saved search conditions and parameters.

Note: OmniFind Analytics Edition server has to be accessible in order to use the bookmark. The bookmark does not save analysis results of views such as graphs and values, and it cannot be used if the server is inaccessible. To save snapshots of the discovered results, use the report function as explained in section titled 5.5.2, “Reports” on page 134.

With the bookmarking facility, you can perform the following tasks:

- ▶ Saving the bookmark:

When the bookmark link is clicked, a dialog box for saving the bookmark to HTML file appears. Specify the destination directory. Close this window after saving the bookmark. Figure 5-35 shows the associated Save As dialog box.

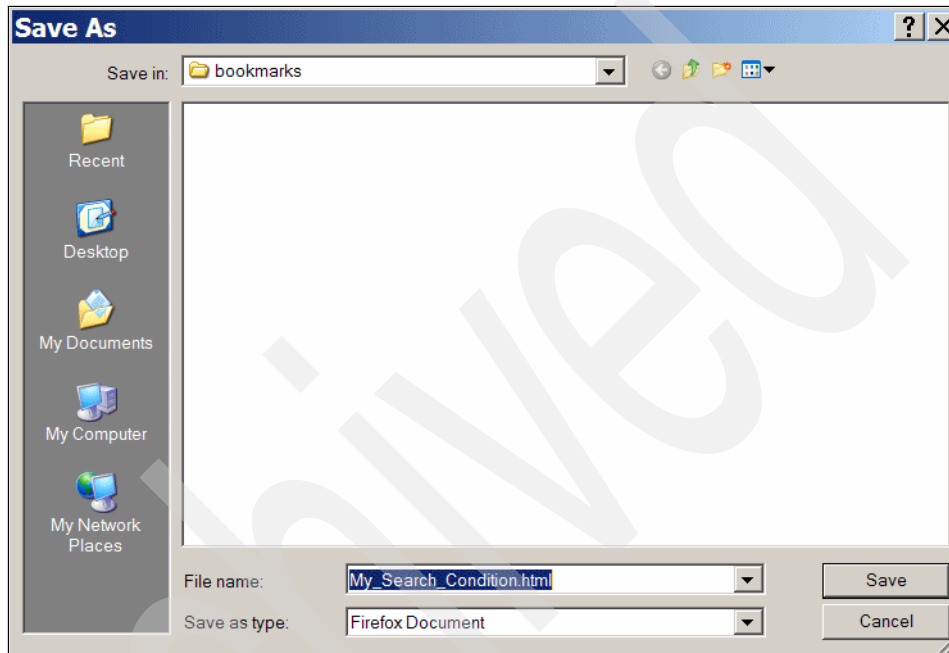


Figure 5-35 Save As option to bookmark search condition

- ▶ Managing the bookmark:

The bookmark can be managed as a local file. It is also possible for users accessing the same Text Miner server to exchange bookmark files through e-mail or to share the files over a network server.

- ▶ Accessing the server from the bookmark:

When a locally saved bookmark is opened, the JUMP button is presented, as shown in Figure 5-36. Click the button to connect to the Text Miner application that is hosted on the OmniFind Analytics Edition server.

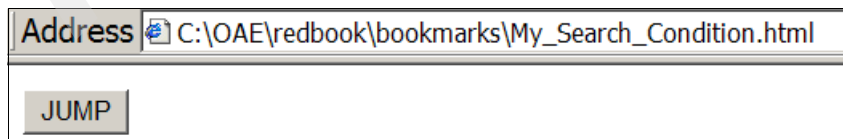


Figure 5-36 Accessing the server from the bookmark

Note: A bookmark saves information contained in the right frame, which includes the search condition, view selection information, view parameters, and vertical and horizontal categories. Meanwhile, the bookmark does not save information in the category tree frame on the left, which is the category sort method and categories that are currently expanded.

Because the bookmark does not save the analysis results, but rather the information required for analyzing a particular set of information, the bookmark can be used repeatedly to analyze results when data has been changed. If a category used as a search condition within the bookmark is deleted, then the search condition becomes invalid. A warning message in red appears for the invalid condition, and the search result becomes 0 hits (“all” if the NOT operator is used in the search).

5.5.2 Reports

The report feature will save the displayed results as a local HTML file. The results of the saved view can be viewed on your local browser, and access to the server is not required for opening the report. To resume analysis with the saved search condition and parameters, the report provides a link to connect to the OmniFind Analytics Edition server.


To overcome the restriction imposed by HTML reports created by the Bookmarks feature, the HTML file created by the Reports feature saves display view, search conditions, parameters required to view the results even when the OmniFind Analytics Edition server is not accessible. The generated HTML file provides a link that connects to the OmniFind Analytics Edition server to continue analysis.

Creating a report

Figure 5-37 shows the report edit window that is generated by clicking the Report link in the Tools view.

In the edit window, you can perform the following tasks:

- ▶ Provide a description and the option to display the description. The description can be up to 2000 characters long.
- ▶ Attach sample documents by clicking the **Attach documents** check boxes.
- ▶ Save the report by clicking the **Create** button (not shown in Figure 5-37). The Save As dialog as shown in Figure 5-38 will be launched for you to enter the report name and location.

Address  http://localhost:9080/TAKMI_MINER/ReportEditServlet

REPORT EDIT

Please identify information to include in the report and click the "OK" button on the bottom of this page.

SEARCH CONDITION Searched documents: 94 documents

KEYWORD: fire 94 docs

VIEWS Category

Display description: ☐

Comment:

My report on category view with "fire" as keyword and "COMPONENT DESCRIPTION" as category.

Vertical category: Phrase Constituent

List: Keywords

Sort: Frequency

Max lines to display: 100

Attach documents	Keywords	Frequency
<input checked="" type="checkbox"/>	Ohio STATE POLICE REPORT	7
<input checked="" type="checkbox"/>	of ... fire	5
<input checked="" type="checkbox"/>	WIRING HARNESS	4
<input checked="" type="checkbox"/>	VEHICLE CAUGHT Ontario FIRE	4
<input checked="" type="checkbox"/>	vehicle fire	4
<input checked="" type="checkbox"/>	for ... client	4
<input checked="" type="checkbox"/>	in ... ENGINE COMPARTMENT	3
<input checked="" type="checkbox"/>	ENGINE COMPARTMENT	3

Figure 5-37 My report on a category view

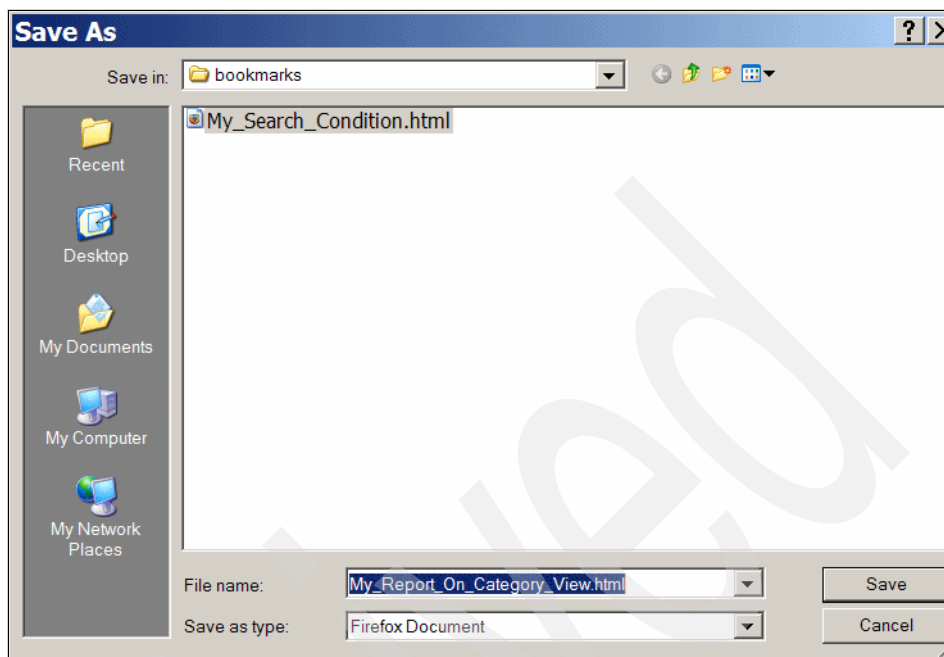


Figure 5-38 Save As option for reports

Viewing report

Use your browser to open the saved report to view the contents of the generated HTML file. The results in the HTML will display the static results that were saved during the report creation time.

To resume analysis, refer to Figure 5-39, where the link labelled **here** can be clicked to reconnect to the server. The link will launch the Text Miner application with the view, search conditions, and parameters preserved in the saved report.

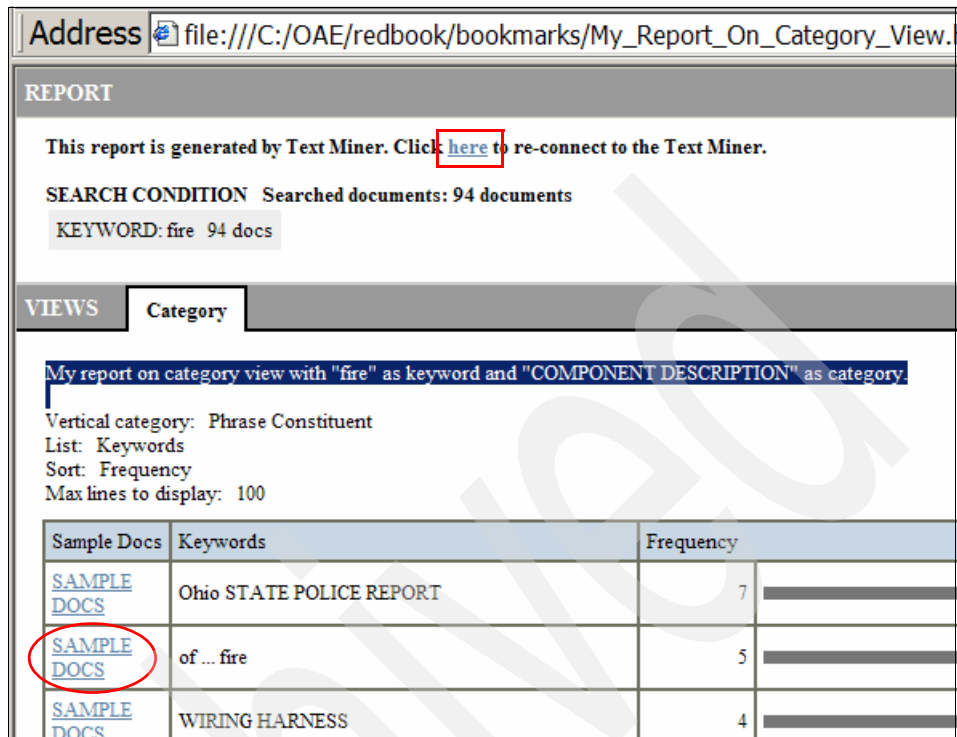


Figure 5-39 Accessing category view from the saved report

Link to an existing report/sample document

In Figure 5-39, the in-file link to the sample document is displayed for the item rows that were selected for "Attach documents" (during report generation). Click the link labelled **SAMPLE DOCS** to jump to the corresponding document. Figure 5-40 shows the SAMPLE DOCS view. The sample document provides a quick short description of the data.

C

Address	file:///C:/OAE/redbook/bookmarks/My_Report_On_Category_View.h
SAMPLE DOCS: of ... fire	
Sample Docs	
ELECTRICAL SHORT UNDER THE DASH FIRE, WIRING HARNESS , VEHICLE WAS PARKED FOR A PE STARTED THE FIRE. THE INSURANCE WAS INVESTIGATING THE CAUSE OF THE FIRE. THE AUTHO MALFUNCTIONED. *YH	
ELECTRICAL SHORT UNDER THE DASH FIRE, WIRING HARNESS , VEHICLE WAS PARKED FOR A PE STARTED THE FIRE. THE INSURANCE WAS INVESTIGATING THE CAUSE OF THE FIRE. THE AUTHO MALFUNCTIONED. *YH	
ENGINE COMPARTMENT FIRE, UNKNOWN SOURCE OF THE FIRE. TT	
AFTER BEING PARKED OVER NIGHT, NEXT MORNING NOTICED DASH BURNED AND THE STEERING	
AFTER BEING PARKED OVER NIGHT, NEXT MORNING NOTICED DASH BURNED AND THE STEERING	
SAMPLE DOCS: WIRING HARNESS	
Sample Docs	
ELECTRICAL SHORT UNDER THE DASH FIRE, WIRING HARNESS , VEHICLE WAS PARKED FOR A PE	

Figure 5-40 Sample documents that highlight the findings

5.5.3 CSV output

The displayed analysis results can be saved as a Comma Separated Value (CSV) file by using the CSV output function. The CSV function will save only the number of results displayed in the view. If the search condition has 210 results but the maximum number of lines to display in the view is set to 20, then only 20 results will be saved in the CSV file.

The generated CSV file is ideal to share with users who do not have access to the Text Miner application but need to view the results in Microsoft's Excel application.

Figure 5-41 shows the Save As dialog box. You can specify the name and the location of the file to be saved in the dialog box.

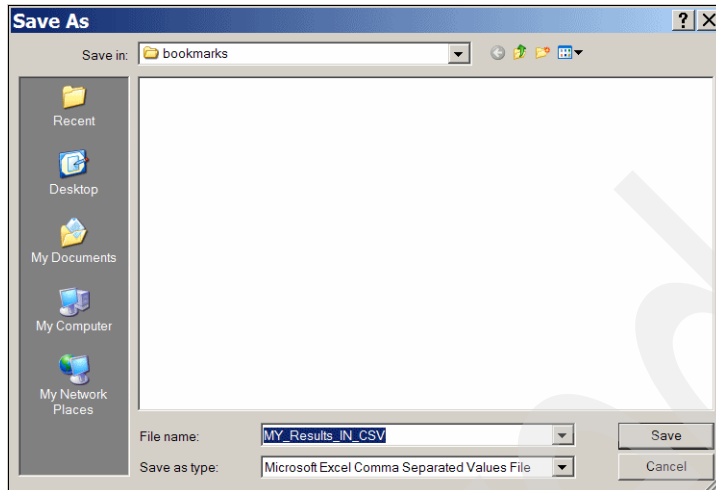


Figure 5-41 Save As dialog to generate CSV file

In Figure 5-42, we show a sample copy of the generated CSV file with Microsoft Excel application.

Microsoft Excel - MY_Results_IN_CSV.csv		
File Edit View Insert Format Tools Data Window Help		
A1 *** Parameters ***		
	A	B
1	*** Parameters ***	
2	Vertical category:	Phrase Constituent
3	List:	Keywords
4	Sort:	Frequency
5		
6	*** View ***	
7	Keywords	Frequency
8	Ohio STATE POLICE REPORT	7
9	of ... fire	5
10	WIRING HARNESS	4
11	VEHICLE CAUGHT Ontario FIRE	4

Figure 5-42 CSV file viewed in Microsoft Excel

This chapter has described the Text Miner application and provided the basics for performing text mining. For more detailed information about text mining, see Chapter 6, “Text Miner advanced” on page 141.

Text Miner advanced

In this chapter, we go beyond the text mining basics and discuss in detail the major methods of analysis in the Text Miner application, the views they are visible in, and some practical applications of these analytic tools.

We cover the following topics:

- ▶ Text analytics versus text mining
- ▶ Frequency
- ▶ Correlation
- ▶ Topicality index
- ▶ Increase indicator
- ▶ Conclusion: Understand what you see

6.1 Text analytics versus text mining

There is a distinction between text mining and text analytics. Text mining is a technology that uses natural language processing to transform unstructured textual data into structured data. While standard text mining ends here, for OmniFind analytics, this is only the beginning. After all the linguistic data is transferred into structured data, OmniFind Analytics takes part in algebraic functions. OmniFind Analytics Edition's Text Miner application applies sophisticated formulae to linguistic data, which results in statistical indexes.

As you learned in Chapter 5, "Text Miner mining basics" on page 87, OmniFind Analytics Edition generates frequency statistics, correlation values for keywords and categories, and Increase Indicator indexes of keywords over time period. When there are anomalies within the dataset, the Text Miner application highlights the specific statistic values in different shades of color, or color underlines, to bring attention to analysts for further analysis.

6.2 Frequency

The frequency of words in a dataset can be powerfully revealing. If we ask ourselves which words should occur with the most frequency in *any* collection of documents, the answer is simply the most frequently occurring words in the language — regardless of the dataset. This is not very revealing. In linguistic terms, these would be grammatical operators, as opposed to lexical entries.

In layman's terms, we mean that the words most frequently used in the language are the ones that work the hardest: those words that hold a sentence together. Among these words are the *helping verbs* (or auxiliary verbs): *do*, *have*, and *be*. Next are *conjunctions*: *and*, *but*, and *or*, which are *coordinators* — they make coordinate clauses and phrases. Of the coordinating conjunctions, *and* is by far the most frequent. There are also subordinating conjunctions, which conjoin a dependent clause with an independent clause. These include *that* and *while*.

This word usage frequency is borne out if we look at the simplest analysis in OmniFind Analytics Edition, in two of our case study databases. The first is the IBM Help Center dataset. In Figure 6-1, where we select Part of Speech and sort them by frequency, the top two most frequently used words are *be*, and *and*.

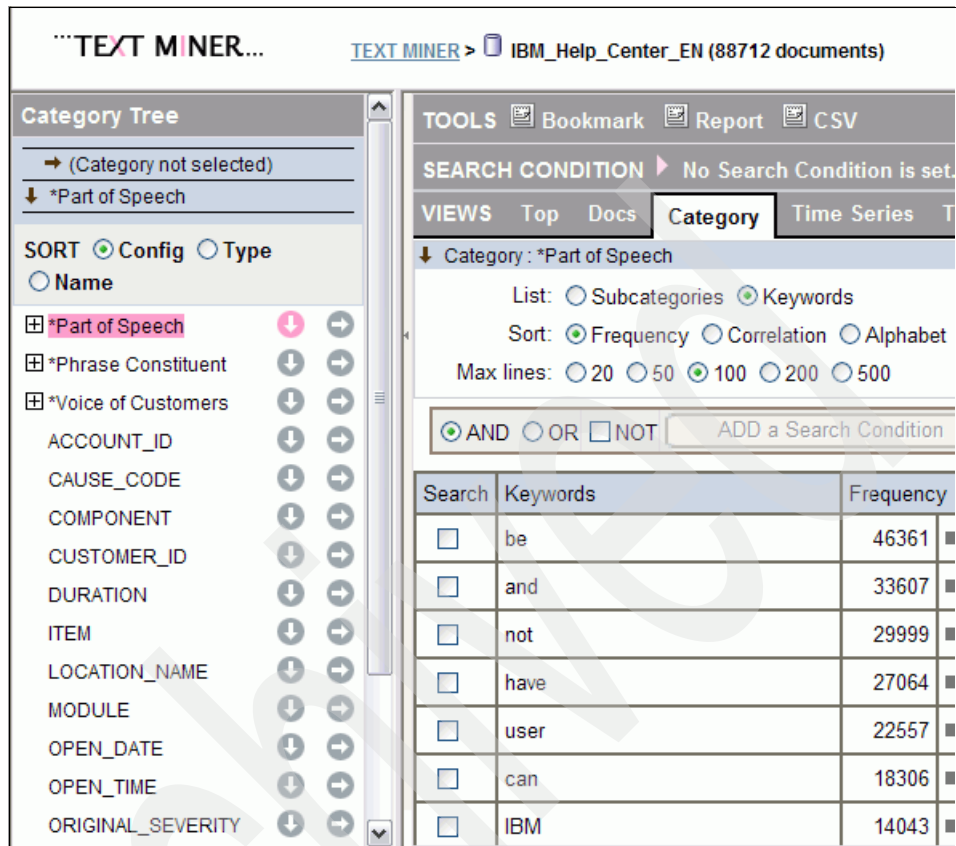


Figure 6-1 High frequency words in IBM Help Center database

We find similar results in the NHTSA document set. See Figure 6-2. The top two most frequently used words in the dataset are *be* and *and*.

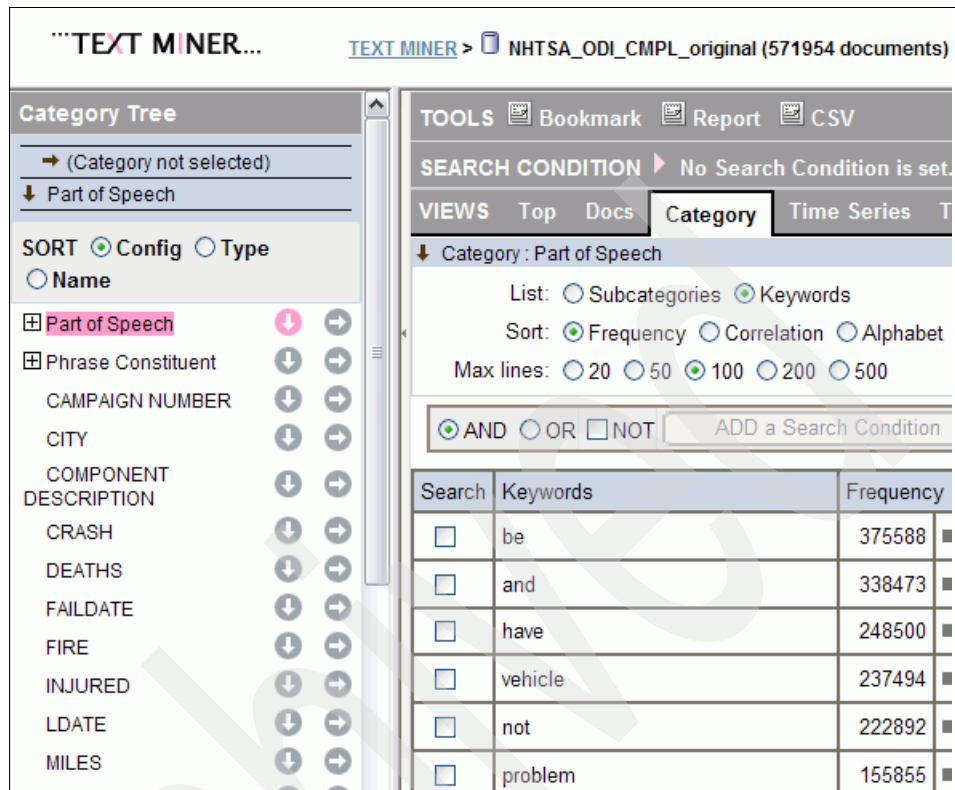


Figure 6-2 High frequency words in the NHTSA database

In addition to the helper verbs and conjunctions, we also see *IBM* as high on the list in the IBM Help Center data corpus. This is not a surprise, given the fact that it is a set of data from IBM customers. Also, the word *not* is high on the list. Considering that most people call the Help Center when they have problems with something, it is not surprising to find the word *not* being one of the most frequently occurred words in the dataset. Likewise, in the NHTSA dataset, the words *vehicle* and *problem* are to be expected, because this is a collection of complaints about vehicles. The point here is that in addition to grammatical words, we will always have to factor in the nature of the data corpus (know the data!).

But we are not interested in the words we expect to see at a high frequency, except in so far as we need to eliminate them. This is what a traditional search engine provides, and what it uses to judge the relevancy of a document: a document is scored in many search engines simply based on the number of times a word occurs in the document. Perhaps, when you were applying for your first position as a Java programmer, you heard of people writing resumes that were literally filled with the word Java. People who understood the recruiters' search bots knew that only the resumes with the highest hit count for the required skills would get a return and hence, a chance to get an interview!

For OmniFind Analytics Edition, frequency should not be the only statistical feature of the tool. Actually, it can be considered the least important analysis tool. The power of OmniFind Analytics Edition lies in advanced statistical analysis. To discover something *unusual*, to *uncover anomalies*, is what the Text Miner is all about! We begin with a discussion of correlation — specifically, correlation as applied in OmniFind Analytics Edition.

6.3 Correlation

Correlation values are used in two of OmniFind Analytics Edition's Text Miner Views: the Category view, and the 2D Map view. We will look at how correlation values are applied, and how to use each of these views to uncover meaning in unstructured data.

6.3.1 Correlation values in Category view: frequency versus correlation

Next we take a look at how the frequency might obscure an interesting trend. To discover the trend, we have to focus on correlation.

The first simple point to make is that correlation requires at least two collections of search results. More simply, the Components category does not correlate with anything by itself — when it is the only condition. In Figure 6-3, notice that when only the Components category is selected, all correlation values are 1.

Part of Speech

Phrase Cons

CAMPAIGN

NUMBER

CITY

COMPONENT DESCRIPTION

CRASH

Components

Customer's V

DEATHS

Exploring

FAILDATE

FIRE

INJURED

No Search Condition is set.

VIEWSTopDocsCategoryTime SeriesTopicDelta2D Map

Search	Keywords	Frequency	Correlation
<input type="checkbox"/>	brake	62693	<div></div> 1.0
<input type="checkbox"/>	engine	52389	<div></div> 1.0
<input type="checkbox"/>	tires	40514	<div></div> 1.0
<input type="checkbox"/>	transmission	33458	<div></div> 1.0
<input type="checkbox"/>	air bag	23636	<div></div> 1.0
<input type="checkbox"/>	wheel	22336	<div></div> 1.0
<input type="checkbox"/>	steering	20312	<div></div> 1.0
<input type="checkbox"/>	seat	20259	<div></div> 1.0

Figure 6-3 Correlation is always 1 if only one condition is set

Now, consider a search of the NHTSA database for fire, intersecting the Components category as shown in Figure 6-4.

TEXT MINER...

TEXT MINER > NHTSA_DB2_20070911 (501376 documents)

Interjection

Noun

Verb

Phrase Constituent

CAMPAIGN

NUMBER

CITY

COMPONENT DESCRIPTION

CRASH

Components

Customer's Voice

DEATHS

Exploring

FAILDATE

FIRE

TOOLS

Bookmark

Report

CSV

SEARCH CONDITION

Searched documents: 17909

Keyword

Searched documents: 17909 documents

KEYWORD: Y(FIRE) 17909 docs DELETE

VIEWS

Top

Docs

Category

Time Series

Topic

Delta

2D Map

Category: Components

List: Subcategories Keywords

Sort: Frequency Correlation Alphabet

Max lines: 20 50 100 200 500

AND OR NOT

ADD a Search Condition

Figure 6-4 Search based on the keyword FIRE and the Components category

The result set is sorted by Frequency. See Figure 6-5.

Searched documents: 17909 documents				
KEYWORD: Y(FIRE) 17909 docs DELETE				
VIEWS	Top	Docs	Category	Time Series Topic Delta 2D Map
Search	Keywords	Frequency	Correlation	
<input type="checkbox"/>	engine	4188	<div></div>	2.2 <div></div>
<input type="checkbox"/>	hood	3090	<div></div>	9.6 <div></div>
<input type="checkbox"/>	under ... hood	1830	<div></div>	14.6 <div></div>
<input type="checkbox"/>	switch	962	<div></div>	2.6 <div></div>
<input type="checkbox"/>	dash	831	<div></div>	4.0 <div></div>
<input type="checkbox"/>	wire	793	<div></div>	4.2 <div></div>
<input type="checkbox"/>	ignition	760	<div></div>	2.6 <div></div>
<input type="checkbox"/>	fuel	746	<div></div>	1.4 <div></div>

Figure 6-5 Keyword: FIRE, Category: Components, sorted by frequency.

Nothing is surprising here: It is not hard to imagine fires occurring in the *engine*, fires being caused by *switches* or *wires*, and fires being started by the *ignition* or *fuel*. Now, by switching the sort to Correlation, we get a different perspective on where fire is started. Figure 6-6 shows the result set sorted by Correlation.

VIEWS	Top	Docs	Category	Time Series Topic Delta 2D Map
Sort: <input type="radio"/> Frequency <input checked="" type="radio"/> Correlation <input type="radio"/> Alphabet				
Max lines: <input type="radio"/> 20 <input type="radio"/> 50 <input checked="" type="radio"/> 100 <input type="radio"/> 200 <input type="radio"/> 500				
<input checked="" type="radio"/> AND <input type="radio"/> OR <input type="checkbox"/> NOT <input type="button" value="Add a Search Condition"/>				
Search	Keywords	Frequency	Correlation	
<input type="checkbox"/>	engine ... catch	324	<div></div>	23.6 <div></div>
<input type="checkbox"/>	under ... hood	1830	<div></div>	14.6 <div></div>
<input type="checkbox"/>	underneath ... hood	211	<div></div>	14.3 <div></div>
<input type="checkbox"/>	from ... hood	162	<div></div>	13.2 <div></div>
<input type="checkbox"/>	cruise control switch	164	<div></div>	12.6 <div></div>
<input type="checkbox"/>	under ... dash	280	<div></div>	10.5 <div></div>
<input type="checkbox"/>	possible ... engine	117	<div></div>	10.4 <div></div>
<input type="checkbox"/>	hood	3090	<div></div>	9.6 <div></div>
<input type="checkbox"/>	in ... ignition switch	143	<div></div>	9.2 <div></div>

Figure 6-6 Keyword: FIRE, Category: Components - sorted by correlation.

While we might have expected the ignition switch to cause fires, we might not have considered the *cruise control switch*. Notice that the cruise control switch has a higher correlation to fire starting than does ignition switch.

In fact, if we exclude the ignition switch from the query, we discover that the occurrences of under hood *and* hood in OmniFind Analytics Edition's initial analysis are also related to the cruise control. For completeness, we look into individual documents for the root cause of fire. Figure 6-7 shows the details of one of the documents. Upon reviewing the actual documents (complaints), we discover that most of the fires are caused by the *speed control cable*, a component of the cruise control switch.

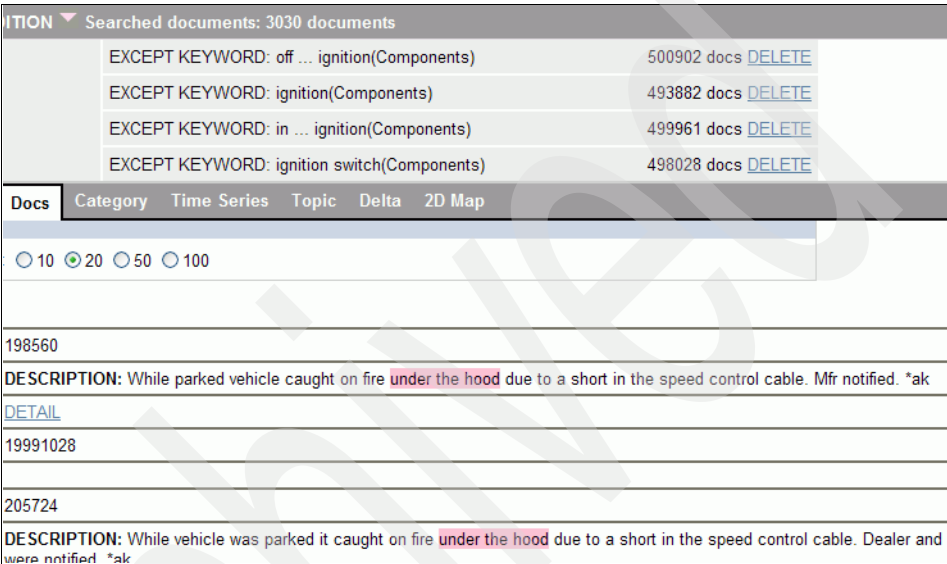


Figure 6-7 The speed control cable was the cause of fires under the hood.

So, the point of this example is that mere frequency analysis would have obscured the cruise control and speed cable completely, and led the analyst to assume that there was nothing unusual in the cause of fires. At most, frequency analysis would lead us to believe that it was the ignition sparking the fuel line. OmniFind Analytics Edition, by using its correlation algorithm, has led to the early detection of a serious problem with the cruise control. Furthermore, OmniFind Analytics Edition has uncovered an even deeper troublesome component, the speed control cable, which is the true cause of fires under the hood.

6.3.2 Correlation values used in the 2D Map view

Let us see how correlation values can be used in the 2D Map view using the IBM PC Help Desk dataset. Figure 6-8 shows the 2D Map view with the Item category keywords as the vertical left-most column (imported from structured data), and Voice of Customers (user-defined category based on unstructured content) as the horizontal row.

We begin with the cell where the cursor is located: the intersection of unable to open and the MAIL keyword.

SEARCH CONDITION ▶ No Search Condition is set.																	
VIEWS	Top	Docs	Category	Time Series	Topic	Delta	2D Map										
Subcategories/ keywords	error	9869	unable to connect	unable to access	unable to open	unable to find	trouble	unable to get	unable to login	unable to log	unable to install	unable to logon	unable to find ... path	unable to send	unable to find path	unable to replicate	unable to boot
		1598	1254	920	778	548	367	343	307	299	291	276	264	261	255	241	221
LEVEL2-3 15855	1136 0.6	23 0.0	25 0.1	13 0.0	37 0.2	20 0.1	17 0.1	3 0.0	5 0.0	0 0.0	7 0.0	0 0.0	4 0.0	0 0.0	1 0.0	0 0.0	15 0.2
APPLICATION 10884	1622 1.3	113 0.5	192 1.0	75 0.5	124 1.0	70 0.8	47 0.7	55 0.9	49 0.9	16 0.2	55 1.1	3 0.0	8 0.1	1 0.0	7 0.1	7 0.1	53 1.4
MAIL 4989	742 1.2	10 0.0	49 2.3	144 2.3	85 0.5	14 0.3	8 0.1	2 0.0	3 0.0	7 0.1	43 1.9	165 9.3	40 1.8	39 1.8	0 0.0	31 1.6	15 0.8
NETWORK 4886	195 0.3	9 0.0	27 0.2	3 0.0	10 0.1	23 0.5	5 0.1	12 0.3	6 0.1	2 0.0	14 0.5	0 0.0	3 0.0	0 0.0	0 0.0	3 0.0	7 0.2
LAPTOP 4267	299 0.5	33 0.3	14 0.1	7 0.0	10 0.1	23 0.5	5 0.1	12 0.3	6 0.1	2 0.0	14 0.5	0 0.0	1 0.0	0 0.0	0 0.0	52 3.2	11 0.5

Figure 6-8 2D Map view of Item and Voice of Customers: Unfavorable categories

The arrow marked (1.) in Figure 6-8 points to the total number of occurrences of MAIL in the dataset. That is, the total number of documents containing the keyword MAIL, in the total dataset, is 4989. Since the total number of documents in the dataset is 88712, then about 5% of the documents contain the word MAIL. This is referred to as the *density* of the keyword.

The number of documents that contain unable to open is 920 (see the arrow marked 2.), making the density about 1%.

Next, we have the intersection of these two collections. The collection of documents that contain the keyword MAIL and unable to open is 144 documents (see the arrow marked 3.). The density of this intersection of collections of documents is about 0.2%. This is a relatively insignificant number. It would not be flagged by traditional search. Yet, OmniFind Analytics Edition has indicated that it *is* significant! In the actual window, this cell is shaded light pink. The correlation value is 2.3 (see the arrow labeled 4. in Figure 6-8). This is because the correlation value is *higher than expected*.

Let us explore the concept of higher than expected in a bit more detail. Look at the number representing the intersection of APPLICATION (5.) and error (6.). The numbers for these two categories are relatively higher than those for MAIL and unable to open, except for the correlation value. The correlation value for

documents containing both APPLICATION and error is significantly less, at only 1.3. This indicates that the co-occurrences of APPLICATION and error are not significant for this dataset. It is normal.

How does this work? Basically, we need to compare the *product of the densities* of collections A and B (multiply the densities), to the *density of the intersection* of A and B, in order to see how the intersection of A and B *deviates* from the norm (the *independence* of A and B). In other words, we want a ratio of all the users who called in about not being able to open their MAIL (combination of unable to open and MAIL) to all the users who called in about MAIL (MAIL), which sets the norm for MAIL. This ratio is the deviation from the norm.

We can represent it in percentages as shown in Figure 6-9.

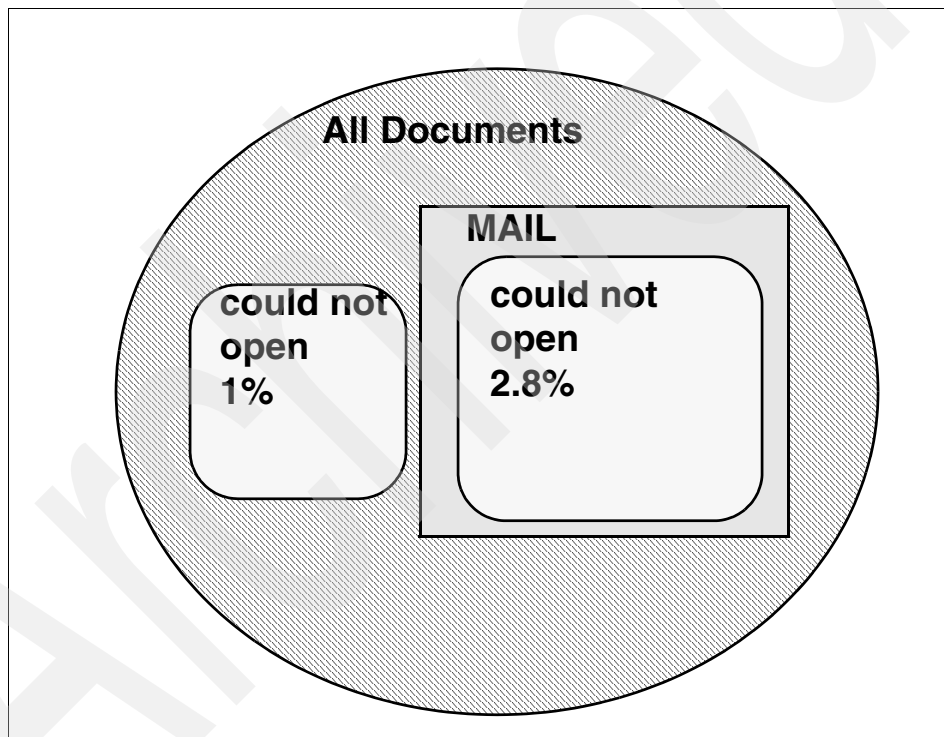


Figure 6-9 Percentage is larger for deviation

The correlation, then is the *ratio* of 2.8 to 1. This is a higher number than what OmniFind Analytics Edition displays in the cell in Figure 6-8 on page 149. In statistics, this is called *reliability correction*. When the number of documents in the intersection of A and B — in our case, the number of documents that contain *both* MAIL and unable to open — is *small* (relatively speaking), the correlation value is *less* reliable.

OmniFind Analytics Edition's Text Miner uses the statistical *interval estimation* to make the correlation number smaller to some degree. The topic of interval estimation is beyond the scope of this book. The bottom line is that OmniFind Analytics Edition reduces the size of the correlation value to some number that it considers to be reliable. In our case, OmniFind Analytics Edition calculated 2.3 as the correlation value.

Finally, let us look at the rest of the “hot” correlations shown in Figure 6-8 on page 149. We call them “hot” because their correlation values have put them on the “heat map” (see 9.2.2, “Setting up Correlation Detection” on page 230). We sort them in Table 6-1 from highest to lowest.

Table 6-1 Correlation values for “hot” problems

Combined keywords and voice of customers	Correlation value
unable to send ... MAIL	9.3
NETWORK ... trouble	5.0
unable to boot ... LAPTOP	3.2
unable to open ... MAIL	2.3

With this simple set of correlation values, we can come to a number of useful generalizations about our PC Help Desk queues. We know, for example, that people have greater difficulty sending their mail than opening it. It is also possible, however, that people are more likely to contact the Help Desk if they cannot send their mail, than if they just cannot open it.

OmniFind Analytics Edition has the texts of all the documents available right in the Text Miner. To do further investigation, we can drill down to the actual messages. Further, if we want to grow our business, we could conclude that our e-mail system might need some attention: Perhaps money spent in improving the ease of sending mail, or, at least investing some money into discovering a pattern as to why people have trouble sending their mail, would alleviate money spent on the PC Help Desk.

Without any difficulty, OmniFind Analytics Edition can facilitate the second of these actions without any more investment! The Text Miner can easily tell us what makes it hard to send e-mails for many users (relatively speaking). If we dig a little, by joining unable to open and EMAIL in a search condition, and having Part of Speech (includes all words and phrases in the dataset) as the other document collection, Figure 6-10 shows the reason for the high correlation values.

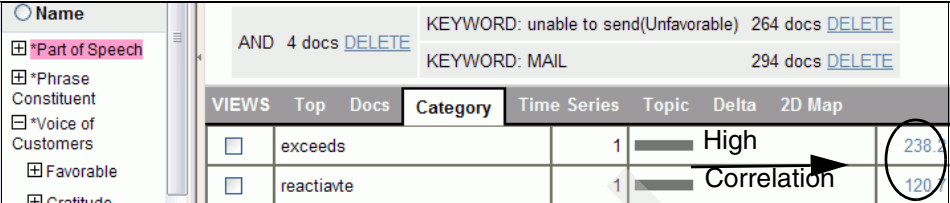


Figure 6-10 Exceeds and reactivate surface as cause of troubles sending E-Mail

Through investigation, we discover that the problem of users not being able to send e-mails is because the users have exceeded their e-mail database size limit. One way to reduce calls related to this matter is either increase the maximum e-mail database size limit, or create a better message to users when this is the reason that they cannot send e-mail. Furthermore, the message should be clearly stated as why the user can no longer send any e-mails out and what specific actions the user should take to reduce the mail database size and then to re-activate the mail database. This will not only reduce the number of calls related to inability to open e-mail, but also calls related to re-activating the mail database.

6.4 Topicality index

The *topicality index* measures *deviation* in a *topic* over *time*. By topic, we mean keywords, subcategories, concepts, and any of the entities that can be annotated in OmniFind Analytics Edition.

The first step is to establish what the norm is for the topic. People all over the world talk about the weather every day. How much do they talk about the weather? What is the average amount of weather talk each day? We need to calculate this as an *average*. We cannot take a sample from any given day or any given week because one particular week might not be representative. It might be the week when everybody was away on vacation.

Now, how does this average weather talk change when there is severe weather, such as a big hurricane? The weather might be so severe that everybody in the world is suddenly talking a lot about the weather. This is what the topicality index is aiming to do: *measure how keywords and subcategories deviate from the average frequency over a specific time line*. Data that shows a higher frequency (of keywords and subcategories) in one time line relative to data for other time lines is highlighted, and normalization is carried out so that data will not be highlighted in response to over-time changes in all searched documents.

6.4.1 Topic view

The Topic view in Text Miner actually shows the Time Series view, in the Frequency row. As an example, we ingested an e-mail database with a metadata field called `isCompleted` that shows whether an e-mail chain had ended, or whether it was still marked for follow-up. Compare the two figures shown in Figure 6-11 and Figure 6-12. The first one, Figure 6-11, shows the Time Series view, and the second one, Figure 6-12, shows the Topic view. The corresponding time line data is indicated in each figure:

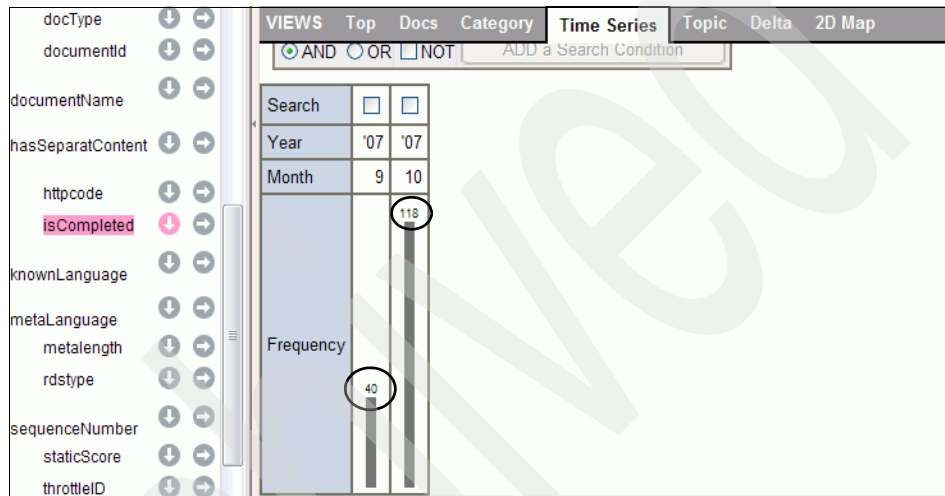


Figure 6-11 Time Series view, average frequency of occurrences of `isCompleted=1` (true)

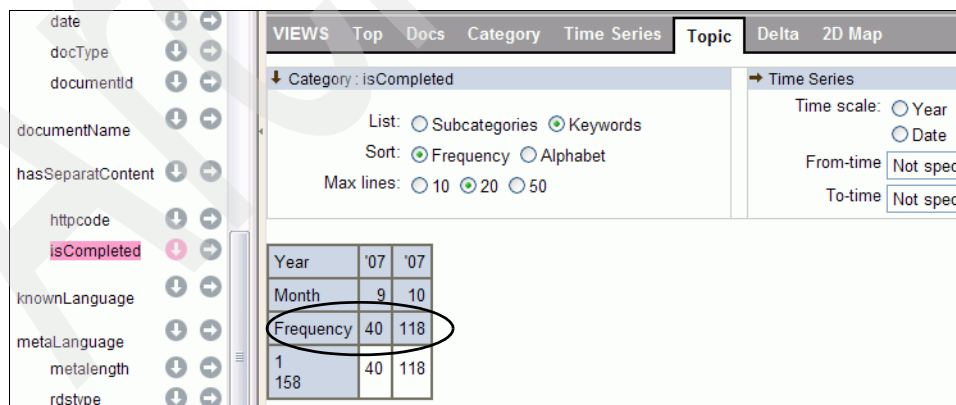


Figure 6-12 Topic view of the same data as in the prior figure

This example shows that the frequency is the actual number of documents. The frequency value is used, but modified, in the topicality index calculation.

As we did with the correlation discussion, we begin by looking at the different views that use the topicality index, and what the numbers mean. Figure 6-13 shows the Topic view for the NHTSA database. It covers the period from 06/2003 to 07/2004, with the category of Complaints as the search condition.

VIEWS	Top	Docs	Category	Time Series	Topic	Delta	2D Map										
Year				'03	'03	'03	'03	'03	'03	'04	'04	'04	'04	'04	'04	'04	
Month	1.			6	7	8	9	10	11	12	1	2	3	4	5	6	7
Frequency				3763	4675	4133	3951	4078	3606	3327	6433	3855	4629	4075	2824	3622	3757
Lack of action or resolution 15139				89	113	145	144	147	142	183	355	187	274	238	177	260	248
Lack of call back 819				6	8	7	8	7	6	5	33	5	9	3	7	10	8
Attitudes 596				8	6	5	7	7	8	7	23	7	11	11	4	11	6
Slow service 570				2	3	9	7	6	5	2	16	9	13	9	2	5	9
Factory defect 397				4	5	5	3	1	5	5	18	2	5	4	2	8	5
Lack of authority 362				5	4	1	6	5	4	2	8	0	1	5	4	6	3

Figure 6-13 Topic view, Complaints category from June 2003 to July 2004

The frequency of *all* complaints is on the Frequency row (marked by arrow 1. above), and it corresponds to the Time Series in Figure 6-14.

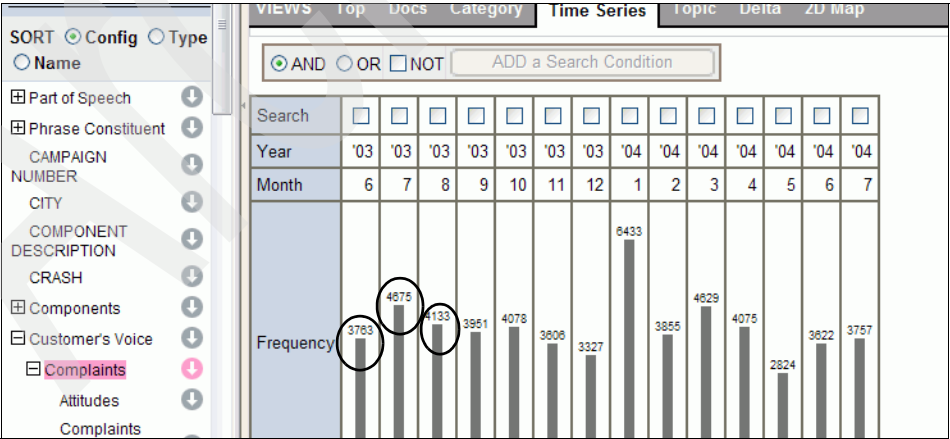


Figure 6-14 Time Series view, Complaints category from June 2003 to July 2004

Focusing on Lack of action or resolution in the first line of the Topic view (Figure 6-13), we see the total number of complaints about Lack of action or resolution is 15139 (marked by arrow 2). Without any more analysis, we know at least that there are a lot of complaints about lack of resolution, which is in itself an important indicator for a company.

If we look at the frequency values for Lack of action or resolution for each month (arrow marked 3), we see that this topic becomes more dense by the month, reaching a peak in January. So, does this mean that support staff grew increasingly lax in their resolution of issues as the time approached January of 2004, and continued to be lax through to July of 2007? In fact, this is exactly what OmniFind Analytics Edition is telling us — Although it is disturbing to the business owners who are implicated, it is totally reliable! It is, however, still just the frequency, not the topicality index.

The topicality index is not represented by a numeric value in the user interface. Instead, it is represented by the *highlighting*. The density (darkness) of the highlighting represent how the keywords and subcategories for a given line on a given date deviate from the average frequency of those keywords and subcategories on all dates. So, out of 3763 complaints in June of 2003, 89 complaints were about lack of action or no resolution. Note that the number 89 is not highlighted at all. This means that the topicality index for Lack of action or resolution for June was 0.

In the same month, there were 8 complaints about Attitude, out of the same number of total complaints, 3763. This cell does have some light highlighting, meaning that the topicality index for Attitude was greater in June of 2003 for than it was for Lack of action or resolution. This is where the true understanding comes. The following statements are true:

- ▶ The topicality index for Lack of action or resolution was lower than the topicality index for Attitude in June of 2003.
- ▶ The total occurrences of Attitude, in the set of total Complaints, was lower than that of Lack of action or resolution.
- ▶ The number of occurrences of Attitude was unusually high for June of 2003 (even though there are only 8 of them, but they have some highlights); the number of occurrences of Lack of resolution or action was normal or below normal for the same month (due to lack of highlights to the number).

There are several points to emphasize in the calculation of the topicality index. The first is that Text Miner calculates the *average daily frequency* of the topic for the entire dataset. This is the baseline to which the *frequency of the topic for a single day* is compared.

The average daily frequency is normalized, before it is introduced into the topicality equation. What does this mean? Consider for a moment why the frequency values marked by the 1 arrow change, in Figure 6-13 on page 154. Recall that these are the total number of complaints in a month. In other words, why are people complaining less in June of 2003, than they are in, say, January of 2004? Maybe people are taking vacations in June, and so, are not sending in so many complaints. Perhaps it is cold in January, so people are staying inside, and they are bored, so they have more time to write complaints or the weather is bad in January and there are more problems and thus complaints! The point is: it could be for any number of reasons. As such, in statistics, such unknowns must be factored out.

If we compare the density of Lack of action or resolution for August of 03 to the same for September 2003, we get:

- ▶ August: 145 complaints out of 4133 total complaints, which is about 3.5%.
- ▶ September: 144 complaints out of 3951 total complaints, which is about 3.6%.

We could not say that there were more complaints about Lack of action or resolution in September, because the density was higher, than there were in August. The reason the density was higher was because there were fewer total complaints. So, even though the total number of complaints must be used in the topicality equation, it must be factored out for the final value. This means that the shading for the topicality index will remain the same, regardless of the (Time Series) Frequency values. This can be seen clearly in Figure 6-15 and Figure 6-16. Both figures represent the same database. The only difference is that 3 weeks of arbitrary data were removed from August of 04 in the second figure (compare the 1. arrows):

Year	03	03	04	04	04	04	04	04	04	04	04	04
Month	11	12	1	2	3	4	5	6	7	8	9	10
Frequency	1	8	8	18	32	30	93	2189	93447	100497	91586	89209
end...issue 108307	0	3	3	2	6	5	16	688	26227	29302	26522	25533
end...contact 80799	0	2	3	1	4	5	10	420	18392	21488	20592	19882
end...information 69164	0	0	2	1	1	2	5	338	16033	18532	17525	16725
end...model 68766	0	0	2	2	2	3	11	455	16687	18521	17064	16019
end...system 60619	0	0	1	2	3	3	10	390	13981	16467	15379	14383
end...option 59832	0	0	1	1	2	3	10	366	13590	16262	15328	14269

Figure 6-15 Full database: Note the highlighting pattern (topicality index)

Year	03	03	04	04	04	04	04	04	04	04	04	04
Month	11	12	1	2	3	4	5	6	7	8	9	10
Frequency	1	8	8	18	32	30	93	2189	93447	55714	91586	89209
end...issue 95220	0	3	3	2	6	5	16	688	26227	16215	26522	25533
end...contact 71132	0	2	3	1	4	5	10	420	18392	11821	20592	19882
end...information 60813	0	0	2	1	1	2	5	338	16033	10181	17525	16725
end...model 60376	0	0	2	2	2	3	11	455	16687	10131	17064	16019
end...system 53285	0	0	1	2	3	3	10	390	13981	9133	15379	14383
end...option 52587	0	0	1	1	2	3	10	366	13590	9017	15328	14269

Figure 6-16 44783 documents removed: Highlighting pattern remains the same

The figures show that the removal of documents, although changing the number of total documents, was factored out of the topicality index calculation.

Now, when the average frequency value (for all complaints) has been normalized in this way, it is subtracted from the individual cell frequency (the number of complaints specifically about Lack of action or resolution). This number is divided by the *variation scale* — the product of a variation constant v , and the normalized average frequency value.

The topicality index has informed us that, indeed, our staff was not providing resolution to customers' problems in the first half of 2004. We will drill down further in the next section, by looking at the Delta view for the same trend. However, we must look at last insight that the Topic view gives us, that the Delta view cannot: The Topic View allows us to look at this particular trend on a day-by-day basis. Figure 6-17 shows the Topic view based on individual day of the week.

ALIGN NUMBER			VIEWS Top Docs Category Time Series Topic Delta 2D Map										
ONENT													
PTION													
H													
onents													
mer's Voice													
plaints													
itudes													
omplaints about													
t on hold													
isagreement													
actory defect													
vider													

Day of week	Sun.	Mon.	Tue.	Wed.	Thu.	Fri.	Sat.
Frequency	19055	95816	102872	96922	89572	77336	19803
Lack of action or resolution 15139	738	2613	2913	2887	2772	2418	798
Lack of call back 819	42	145	145	147	146	123	71
Attitudes 596	44	100	98	101	103	94	56
Slow service 570	26	90	107	107	108	86	46
Factory defect	30	56	50	79	83	75	24

Figure 6-17 Topic view - Support deteriorates on the weekends

We do not want to confuse this image with what we had established earlier: There was a *specific* problem in the first half of 2004. Now we are looking at the day-to-day trend. We are looking at topicality for the whole dataset, for any and all weeks; this is a *generality* of our support system: It is apparent that our support staffs' inclination to action deteriorates as we head into Thursdays and Fridays, and then it is the worst on Saturdays and Sundays.

6.5 Increase indicator

The *Increase Indicator* index measures the increase in the frequency of use of keywords and subcategories along the time line. In addition, it is an indicator of the *future* increase trends of the keywords and subcategories. This is achieved by measuring how much the frequency on each given date deviates from a constant frequency. Of course, the frequency is never actually constant — it is a statistical assumption that the past time series frequency is constant. This is

referred to as *constant noise*, in the frequency time line, and is estimated using the Poisson distribution¹. With this scale, the variation is calculated in terms of a scaling factor. Finally, it is normalized in the same way we discussed before, to eliminate the frequency variations over time, as a factor.

6.5.1 Increase indicator index in the Delta view

The increase indicator index is used in both the Text Miner and the Alert applications. In Text Miner, the values used in calculating the index are not configurable. They are configurable, however, in the Alert application (see Chapter 9, “Alerting System” on page 221).

Through discussion here, we will help you understand when the Text Miner Increase Indicator index is sufficient for your analysis, and when you would need to turn to the Alert application. Also, this discussion is intended to help you to understand what effect the values have in the calculation of the index.

We begin by looking at a familiar example in the Delta view, and pointing out what all the numbers in the user interface mean. See Figure 6-18.

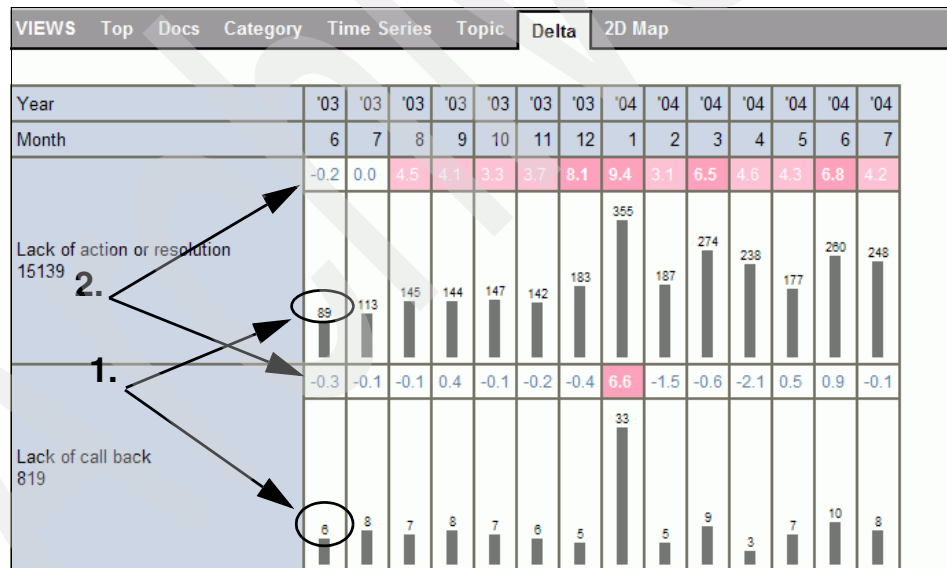


Figure 6-18 The Delta view

¹ Papoulis 1984, pp. 101 and 554; Pfeiffer and Schum 1973, p. 200

As we have seen before, the bar graph is the time series frequency for the keyword or subcategory, with the values (arrow 1.) indicated above the bars. The increase indexes are numbers in the boxes (arrow 2.), with some of them highlighted, along the top of the cells.

In calculating our index, we need the average global frequency — that is the average frequency of all searched documents — over time. We also need the average keyword frequency over time.

Note: The span of the time series — over days, weeks, months, or years, is relevant to the increase index. These can be set for the Alert application (see 9.2.1, “Setting up Increase Detection” on page 223).

In simple terms, to get an average, you add all the frequencies together and divide by the number of dates in the series. However, this is too simplistic. You have to account for the decay factor. *Decay* means that the past becomes less and less relevant, the more distant it gets. Each of the frequencies is weighted, then according to a decay constant. So, it is not just adding all the frequencies together, as equally contributing values in the average calculation; instead, the decay factor is applied to each frequency first. Thus, if the decay value is 0.85 (this is the default for Text Miner’s increase index), the $(n - 4)^{\text{th}}$ date’s frequency contributes less — about half — to the calculation as the n^{th} date’s frequency. Both the global and the keyword time series average frequencies are weighted in this way.

Note: The decay value can be set in the Alert application, according to how little you want the distant past to contribute to the frequency average (see 9.2.1, “Setting up Increase Detection” on page 223 for information about how to set it up).

With the weighted keyword time series, we can accurately estimate the frequency for a future date (any date, actually), assuming that the time series is constant, and factor in the variation within all searched documents, using the weighted global average frequency. This estimate gives us the ability to collate the increase index — the degree to which this keyword has increased or will increase — for the particular date. These are the indexes you see in the shaded boxes at the top of the cells (arrow 2, in Figure 6-18 on page 159).

From the Delta view shown in Figure 6-18 on page 159, we see that there is a strong *tendency* for increase in the Lack of action or resolution. This tendency becomes dangerously strong in August of 2003. This is when the cell shading begins. It might be difficult to see the value in this, because we have the data for September of 2003 right in front of us — so we know that, just by looking at the actual frequency numbers, the incidence of lack of action complaints did

We now can construct a *story* around these images of our NHTSA data. Remember that the Delta view is showing us trends towards increase in frequency. In this light, it seems that there was a serious, consistent trend in lack of resolution of the customers' complaints, beginning around August of 2003, and continuing pretty steadily throughout 2004. Since this trend is fairly solid, we can say that there was a general increase in lack of action complaints through the period.

The lack of call back seems, on the other hand, to be pretty isolated — really only strong, quite strong at an index of 6.6, in January of 2004. So, while people were not getting satisfactory resolution to their problems throughout the period, they *weren't even getting their phone calls returned* in January!

Slow service and bad attitude are vague in terms of trends, with the exception that, once again, customers complain about *really* bad attitudes and *really* slow service in January of 2004. The picture is becoming pretty vivid by now: the automotive industry is clearly not able to resolve their customers' issues in January of 2004.

We get some insight as to the reason why in the last figure. It would seem that there is a spike in the customers' use of *defect*, showing up strongly in January. Now, the interesting thing here is that the increase indexes actually give us clues that the complaints about defects really began to increase much earlier than January of 2004.

6.6 Conclusion: Understand what you see

While all of this is intriguing, we must be careful to understand what these images *do* tell us, and what they *do not*. They *do* tell us that, for some reason, automotive customers were greatly dissatisfied with the response of the industry in 2003 to 2004. The data images also *do* tell us that the customers perceived suddenly very strongly, or were told quite frequently by their service people, that their problems were due to factory defects, in January of 2004. With this much insight, the industry should take appropriate action. The industry has a path for investigation.

What the data does *not* tell us, at least as far down as we have drilled, is that there was a lack of resolution, or that there were defects, in fact. Recall that the NHTSA data we are using is from the Complaints database, and thus is collected customer input — so, we *do* know how many complaints there were, and what those complaints were about.

We do *not* know, however, that the complaints were valid in terms of factory defects. But, we could find out.

One of the strengths of OmniFind Analytics Edition, remember, is its ability to combine unstructured data, with structured data, for use in statistical analysis. Thus, if OmniFind Analytics Edition found the correlation between customers' complaints and the actual structured recall data, in the cases of strong correlation, the customers' complaints would be validated.

OmniFind Analytics Edition can provide infinite insight, but it requires human analysts to incorporate both imagination and accurate deductive and analytical reasoning skills. OmniFind Analytics Edition provides the numbers. However, you must understand what you see.

Customizing the dictionary

This chapter describes how to customize the OmniFind Analytics Edition dictionary using the Dictionary Editor application. The Dictionary Editor is a Web application that is bundled with OmniFind Analytics Edition. You can use the Dictionary Editor to customize how data is to be categorized, indexed, viewed, and analyzed.

We cover the following topics:

- ▶ Dictionary Editor overview
- ▶ Working with categories and the category tree
- ▶ Working with keywords and synonyms

7.1 Dictionary Editor overview

The OmniFind Analytics Edition Dictionary Editor is used to customize OmniFind Analytic Edition dictionary. It is built on the concept that you can use categories from a metadata list. With these categories, keywords and synonyms can be added and combined to add a greater meaning to the results of dataset. This is the basis of analysis, which gives users the method for logical data discovery.

OmniFind Analytics Edition Dictionary consists of the following items:

- ▶ Categories organized by the category tree
- ▶ Keywords
- ▶ Synonyms

The Dictionary Editor can be used to add, edit, or delete the items in the Dictionary. You can also use the Dictionary Editor to register keywords within a category, and create one or more synonyms for use with one or more keywords.

7.1.1 Launching the Dictionary Editor

To launch the Dictionary Editor, enter the following URL in a Web browser:

`http://<Server Name>:<Host Name>/TAKMI_DIC`

Figure 7-1 shows the home page of the Dictionary Editor. By default, there are two navigable links, **Select Database** and **Help**.

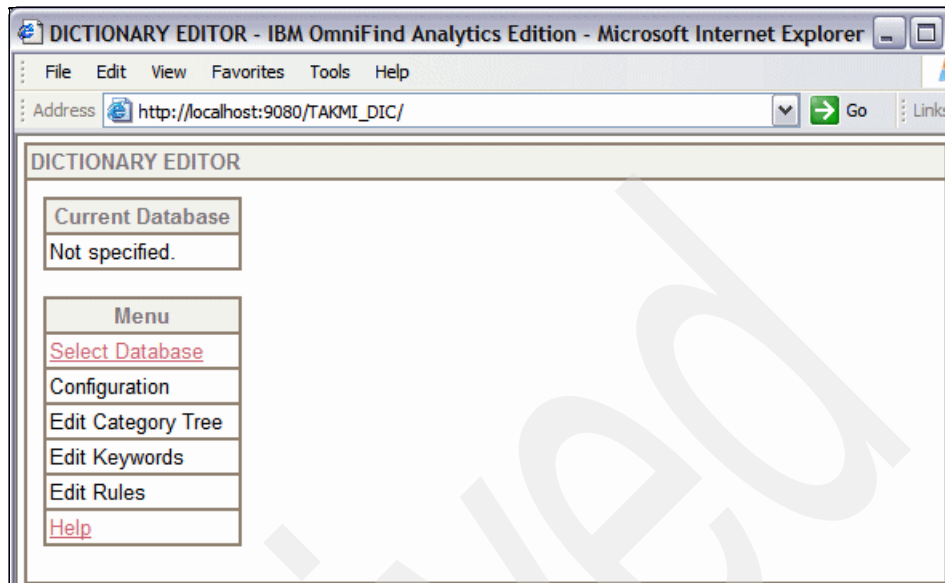


Figure 7-1 Default Dictionary Editor window

7.2 Working with categories and the category tree

Categories represent the keywords in the source data that can be used for analysis. There are standard item categories, system categories, and user defined categories. The standard item categories represent the structured data from the source dataset. The system categories represent categories, such as parts of speech (for example, verbs and nouns) that OmniFind Analytics identified for text analysis. They are extracted from the unstructured text content in the source data. The user defined categories are the ones you defined through the Dictionary Editor. They are also extracted from the unstructured text content in the source data.

Categories can be divided into subcategories. For example, you can have a category based on tires (Tires) in our NHTSA sample database. After further research, we want to isolate tire-related complaints by their manufactures. For this reason, we add a subcategory, Manufacture Name under the Tire category for more detailed analysis.

Adding the categories and subcategories helps analysts to get more meaning from the dataset. A *category tree* is a collection of all categories defined in OmniFind Analytics Edition. There is a virtual root category that functions as the parent category for all categories.

7.2.1 Editing the category tree

To add, edit, and delete categories, you have to edit the category tree using the Dictionary Editor.

Note: For data integrity reasons, you cannot edit the category tree and the keywords at the same time. Even with multiple users and multiple instances of the Dictionary Editor, OmniFind Analytics Edition will not allow you to do that.

As an example, we show you how to create a category tree for the NHTSA sample data. We want to create the following category tree:

- ▶ Automobile
 - Manufacture Name
 - Model Name
 - Model Year
- ▶ Brakes
- ▶ Tires
 - Manufacturer Name
 - Manufacture Year

To edit the category tree, follow these steps:

1. Launch the Dictionary Editor using the following URL in a Web browser:
`http://<Server Name>:<Host Name>/TAKMI_DIC`
2. Click the **Select Database** link that is shown in Figure 7-2. Select a database that you want to work with from the drop-down box, then click **OK**.

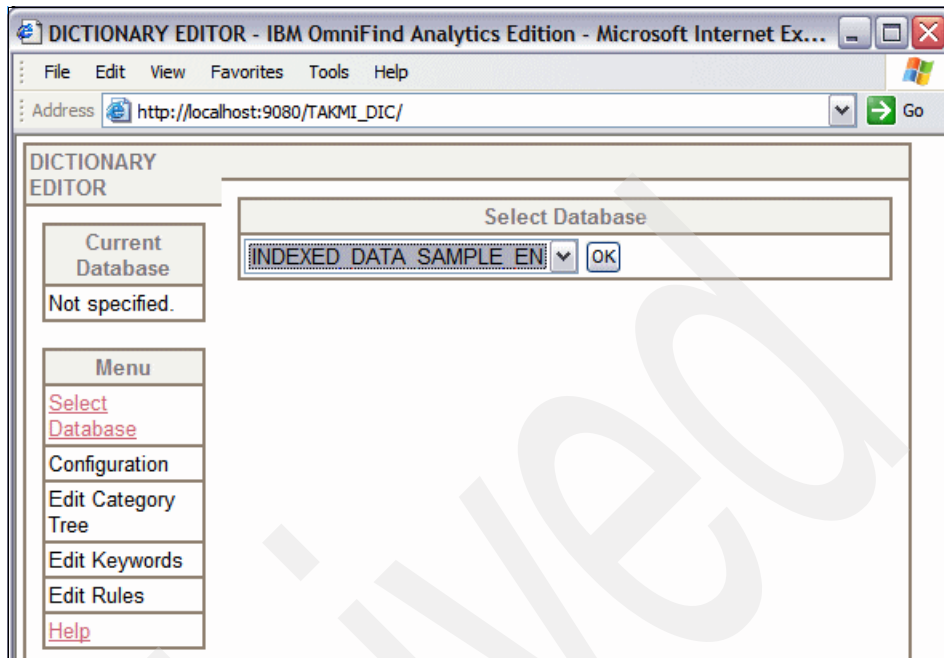


Figure 7-2 Select Database window

Figure 7-3 shows that the database is loaded. All the links under the Menu section are now activated. You can configure, edit category tree, edit keywords, and edit rules from this window. When you are done with each task (such as editing category tree or editing keywords), you need to select the database again to reload the database and proceed with your next task.

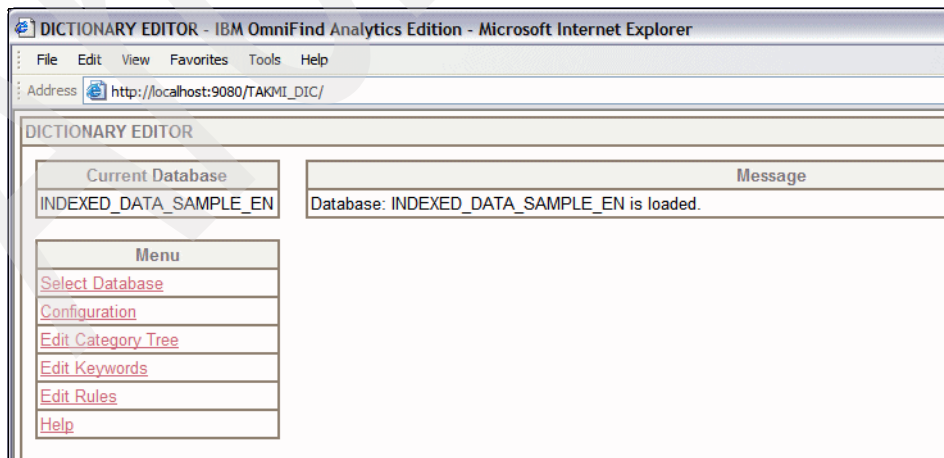


Figure 7-3 Dictionary Editor: Database is loaded

3. Select the **Edit Category Tree** link. When you first start editing the category tree, you should have an empty window with the Add Root Category link. See Figure 7-4.

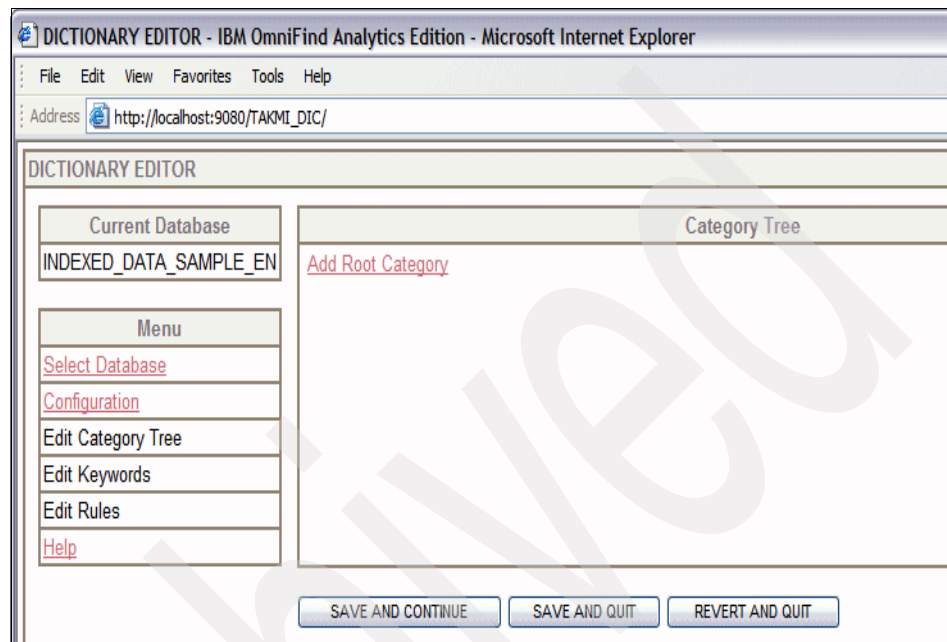


Figure 7-4 Empty category tree

4. Click **Add Root Category**. A dialog box will prompt you to enter the category name. See Figure 7-5.

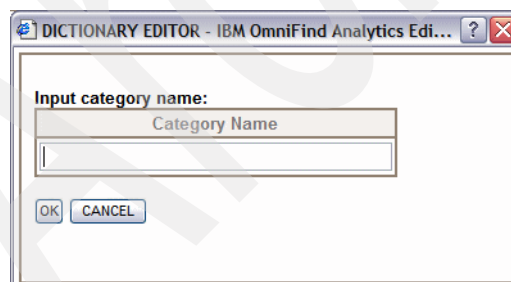


Figure 7-5 Dialog box for entering category name

5. Enter a category name. For our example, we enter Automobile as shown in Figure 7-6. Click **OK**.

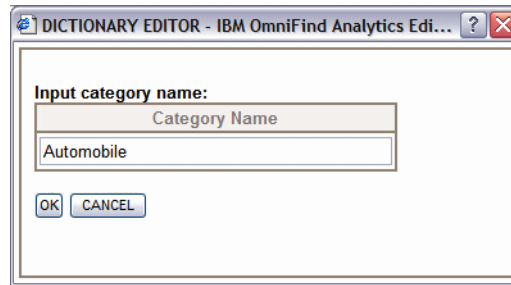


Figure 7-6 Create the Automobile category

The category tree will now reflect the new category you just added. See Figure 7-7.

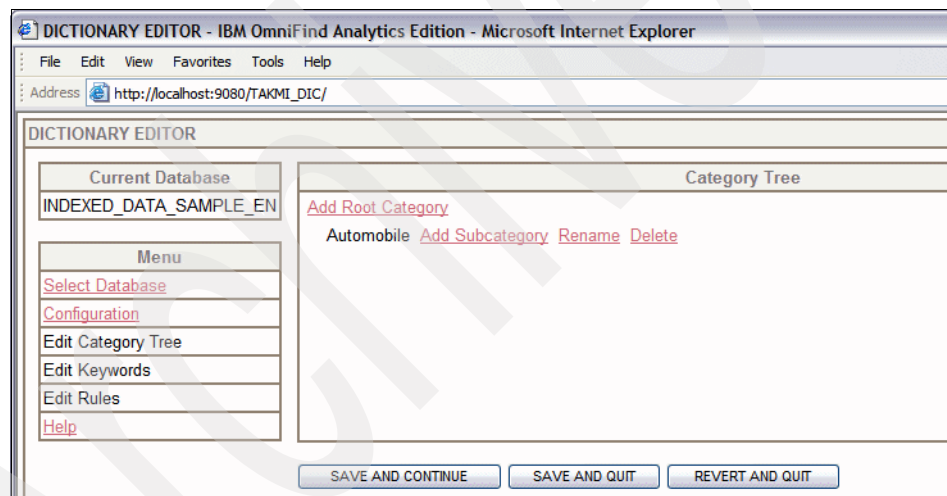


Figure 7-7 Category tree with Automobile category added

Note: In the Category Tree, you can perform the following task:

- ▶ Add a category under the root category by clicking the **Add Root Category** link.
- ▶ Add a subcategory under an existing category by clicking the **Add Subcategory** link right next to the existing category.
- ▶ Rename a category by clicking the **Rename** link next to the category.
- ▶ Delete a category by clicking the **Delete** link next to the category.

6. Add appropriate subcategories. For our example, we want to add the following subcategories under the Automobile category:

- Manufacture Name
- Model Name
- Model Year

To do so:

- Click **Add Subcategory** next to Automobile.
- Enter the subcategory name (Manufacture Name), and click **OK**.
- Repeat the above two steps for Model Name and Model Year.

Remember to always click the **Add Subcategory** link next to the existing category (Automobile) to create its subcategories. Figure 7-8 shows the category tree after we added the Automobile category and its subcategories.

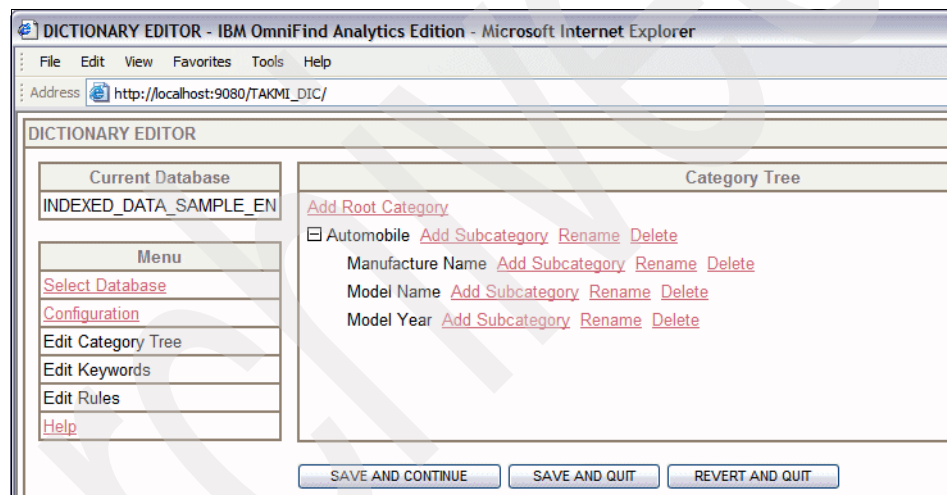


Figure 7-8 Category tree with Automobile category and its subcategories added

Note that the links next to the categories are the actions available specific to that category. So, if you want to create subcategories under Manufacture Name, you should click the **Add Subcategory** link next to it. If you want to add subcategories under Model Year, then you click the **Add Subcategory** link next to the Model Year. This is the same for the **Rename** and **Delete** link. If you made a mistake, you can always rename the category or delete it and re-create it again.

7. Add additional categories under the root category. For our example, we have to add:

- Brakes
- Tires

To do so:

- Click **Add Root Category**.
- Enter the category name (Brakes), and click **OK**.
- Repeat the above two steps for Tires.

Figure 7-9 shows what we have so far, with the additional Brakes and Tires categories added.

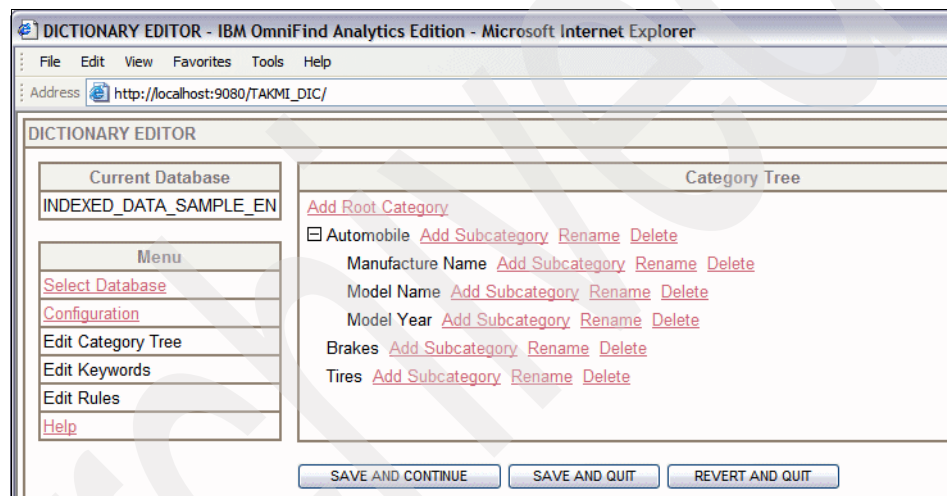


Figure 7-9 Category tree with Brakes and Tires categories added

8. Add additional subcategories if any to the category tree. For our example, we have to add the subcategories to Tires as follows:

- Manufacturer Name
 - Manufacturer Year

To do so:

- Click the **Add Subcategory** link next to the existing category (Tires).
- Enter the subcategory name (Manufacture Name), and click **OK**.
- Click the **Add Subcategory** link next to Manufacture Name.
- Enter the subcategory name (Manufacture Year), and click **OK**.

Figure 7-10 shows what we have in the category tree.

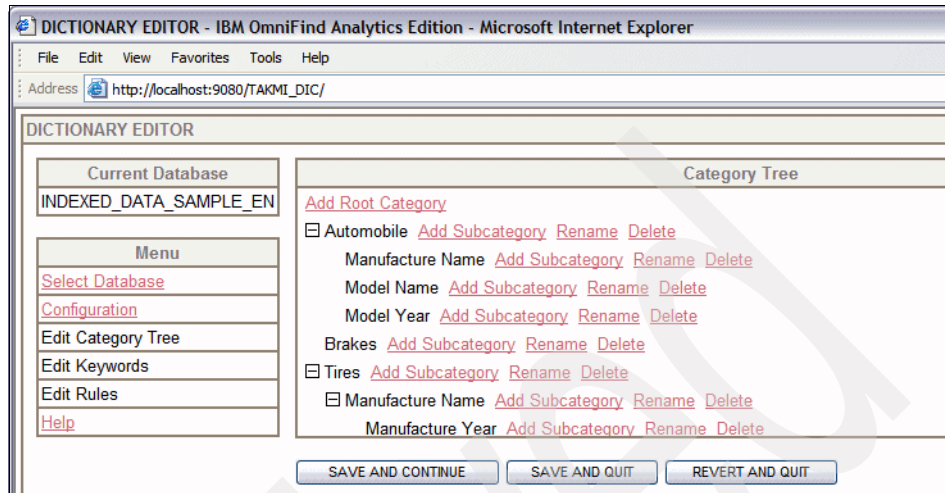


Figure 7-10 Category tree with subcategories of Tires added

9. When you are done with editing the category tree, click **SAVE AND QUIT**. The window will show that the category tree has been updated.

If you want to go back and edit the category tree or edit keywords or perform other actions in the Dictionary Editor, you must reload the database by using the Select Database link.

7.2.2 Configuring the user interface

This is an optional step. To configure the user interface:

1. Launch the Dictionary Editor.
2. Select the database you want to work with.
3. Click the **Configuration** link. The window shown in Figure 7-11 displays.

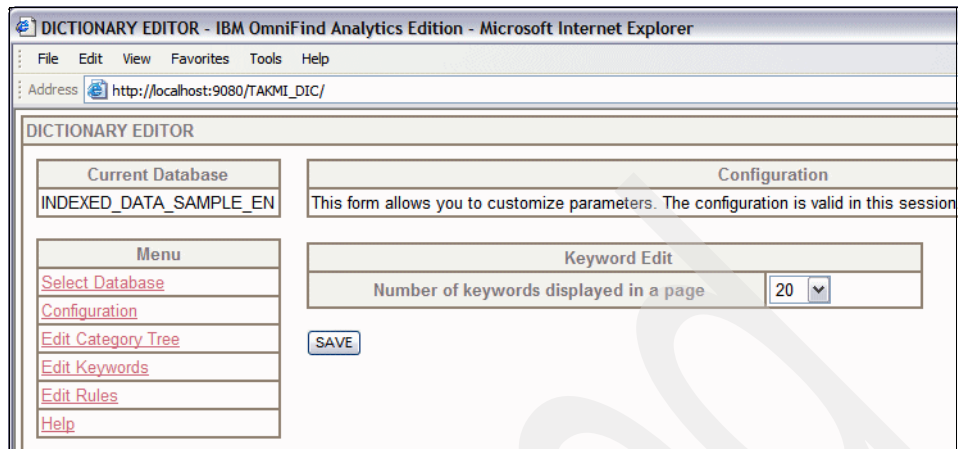


Figure 7-11 Dictionary Editor: Configuration window

4. Currently, there is only one option you can configure in this window, the Number of keywords to be displayed in a page. Select the **Number of keywords to be displayed in a page** from the drop-down button. See Figure 7-12 on page 175.
5. Click **Save** to save your changes.

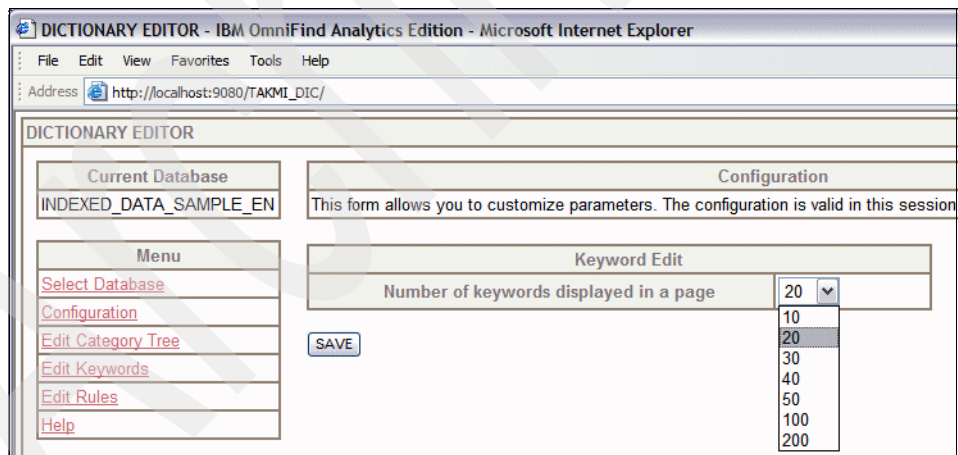


Figure 7-12 Dictionary Editor: Set number of keywords displayed in a page

7.3 Working with keywords and synonyms

OmniFind Analytics Edition comes with a dictionary for extracting nouns and dependency. By using the dictionary, the natural language processing extracts nouns, verbs, and dependency. Specific keywords can also be registered in the dictionary. OmniFind Analytics Edition natural language processing extracts keywords from unstructured source data to provide more insights to the data. Examples of keywords include `cruise control` and `airbag`. Keywords also come with synonyms. The Dictionary Editor allows you to customize the dictionary, to add, edit, and delete keyword and associate synonyms to keywords.

Every keyword belongs to a category. The same keywords or synonyms that belong to different categories have different meanings. For example, the keyword `AC` in the `Component` category means the `air conditioning component` of a car. The same keyword `AC` in the `Power Specification` category means `alternate current`.

Figure 7-13 shows documents that reflect a search on the following keywords:

same ... problem
dealer
CA

In this example, we want to find out complaints to dealers about something that has the same problems as before and that happened in the state of California (CA). So, we search on these three keywords and the highlighted terms reflect the search results.

	KEYWORD: same ... problem(Adjective Noun)	16403 docs	DELETE
AND 138 docs DELETE	KEYWORD: dealer(Proper Noun)	44542 docs	DELETE
	KEYWORD: CA(STATE)	69909 docs	DELETE

Figure 7-13 Keyword example with documents

There are two types of the files that are associated with keywords, the candidate files and the keyword files. The *candidate files* list the keywords and synonyms extracted from your dataset. The keywords in these files are potential keywords (thus candidate) to be used by natural language processor to extra information from your dataset. Candidate files can be generated and imported in CSV format.

For our NHSTA dataset, we have parts.csv and parts.synonym.csv files that contain car component names and their synonyms. They are located at the <TAKMI_HOME>\databases\<database name>\dic\candidate directory.

The keyword files contain the actual keywords that natural language processor use to go through the dataset and create statistical indexes. You can use the Dictionary Editor to edit the candidate files and keyword files.

7.3.1 Editing keywords

Our goal here is to create keywords or register existing keywords to be used by the natural language process. You can get these keywords from the candidate files or you can create them from scratch. To create these keywords in the dictionary, follow these steps:

1. Go to the keyword editing window:
 - a. Launch the Dictionary Editor if you have not launched it already.
 - b. Select the database. You must do this after you finish editing the category.
 - c. Select the **Edit Keyword** link. Figure 7-14 displays.

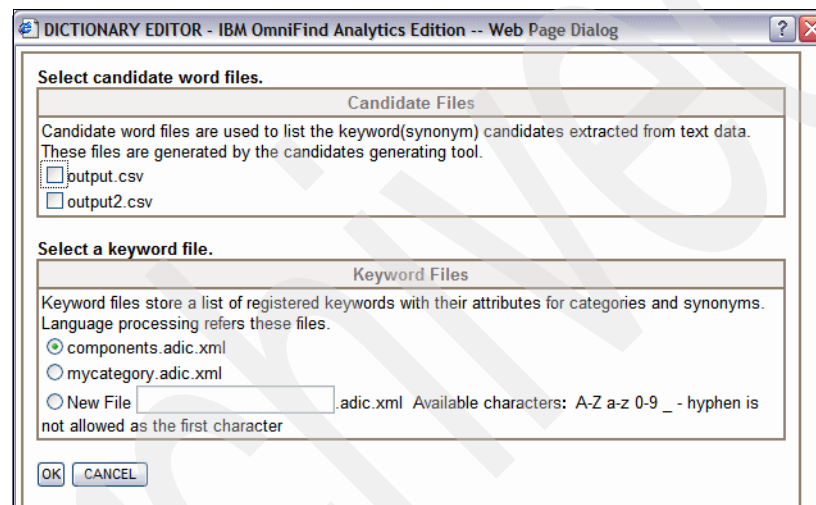


Figure 7-14 Candidate files and keyword files example

- d. Select the candidate and keyword files you want to work with. Click **OK**.

Note that candidate files provide the potential keywords that you want to register in the dictionary to be used by the natural language processor. For a keyword file, you can edit an existing one or create a new one by entering a file name next to the New File field.

For our example, we do not select any candidate file, and we select the **component.adic.xml** file.

Figure 7-15 shows the user interface where you can edit keywords.

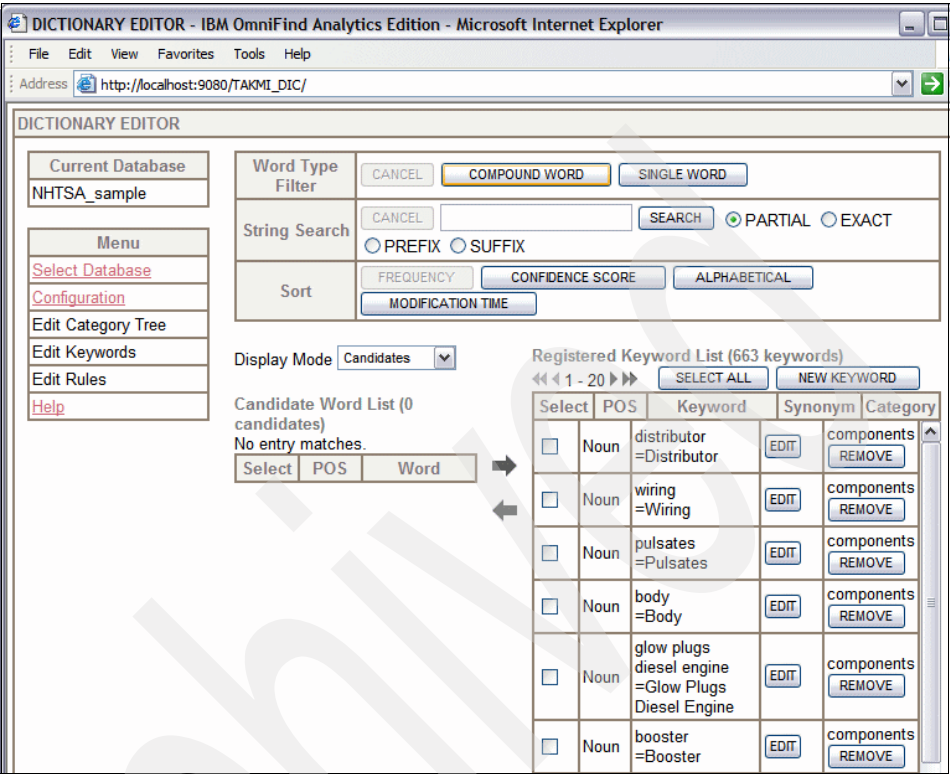


Figure 7-15 Keyword editing user interface

2. Create a new keyword:
 - a. Click **NEW KEYWORD** above the table where the keywords are listed.
 - b. Enter a keyword name. For our example, we enter `cruise control`. See Figure 7-16.

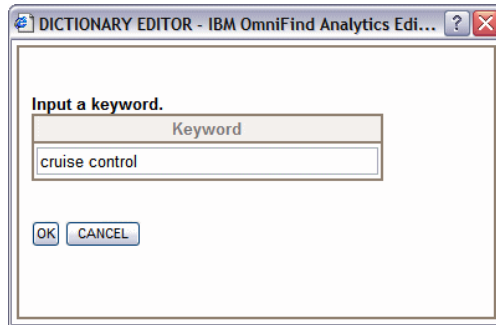


Figure 7-16

c. Click **OK**.

The window now reflects the new keyword you just added. See Figure 7-17.

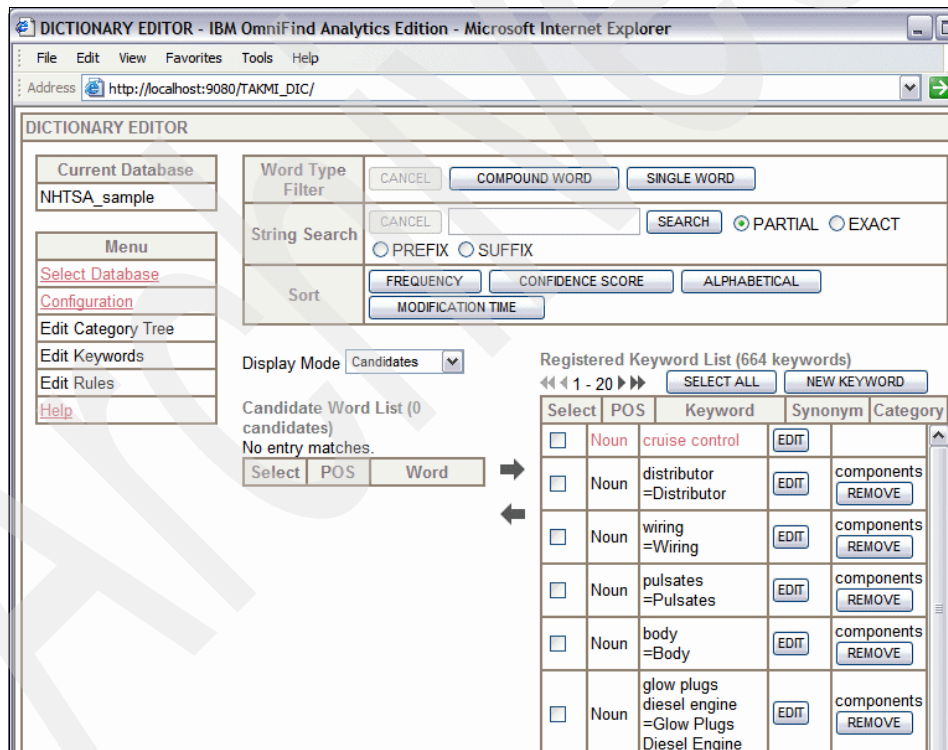


Figure 7-17 Keyword editing interface showing the new cruise control keyword

3. You can create a synonym or multiple synonyms for the keyword. For our example, we will create Cruise Control as the synonym of the word cruise control. Follow these steps to create a synonym for a keyword:
 - a. Click **EDIT** next to the keyword, in our case cruise control. Figure 7-18 displays.

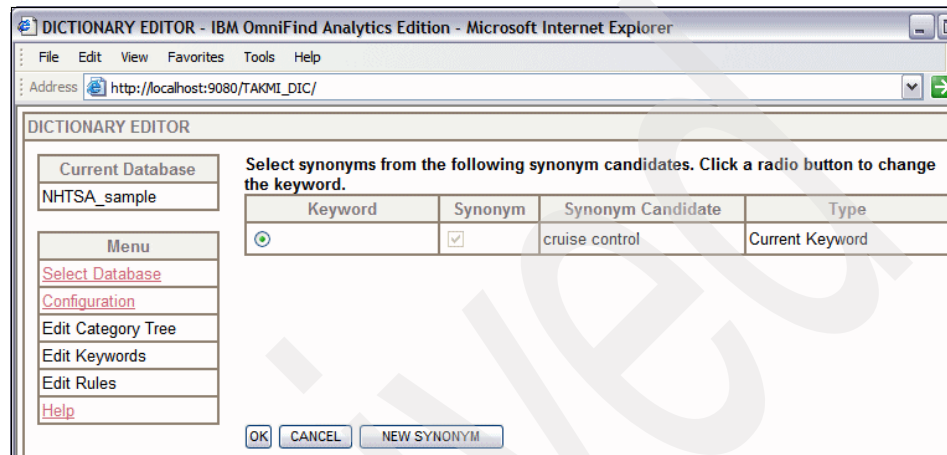


Figure 7-18 Creating a synonym for a keyword

- b. Click **NEW SYNONYM**.
 - c. Enter a new synonym for the keyword. For our example, we enter Cruise Control. See Figure 7-19.

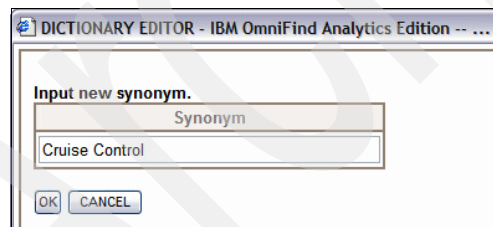


Figure 7-19 New synonym for keyword

- d. Click **OK**. The window should now show the original keyword along with its new synonym. See Figure 7-20.

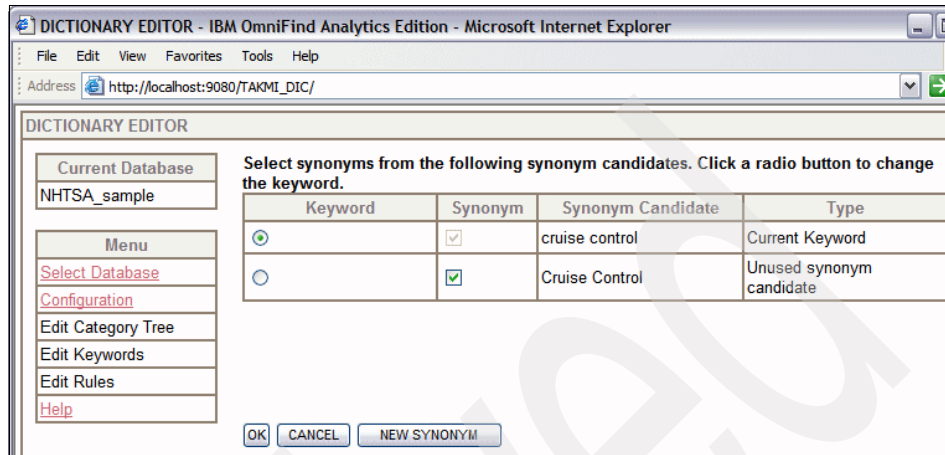


Figure 7-20 New synonym displays next to the original keyword

Notice that the first entry is selected as keyword. You can create multiple synonyms and select a new synonym as the keyword instead of the original one.

- e. Select the check-box to both entries. Click **OK**.

In the keyword editing user interface, the synonym is displayed. For our example, see Figure 7-21.

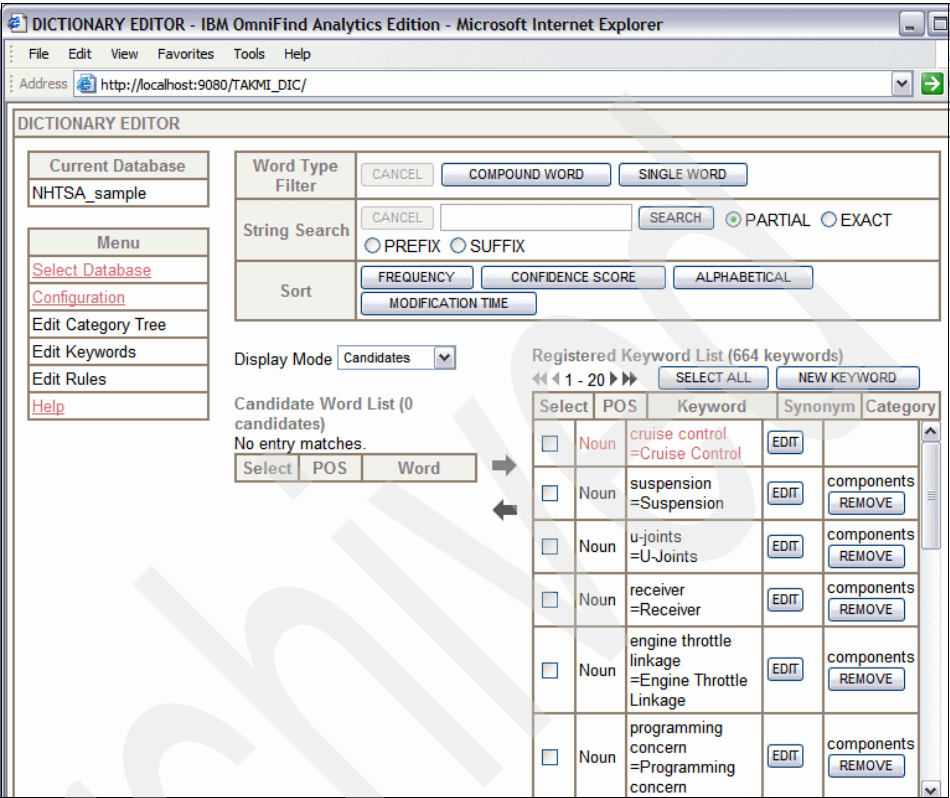


Figure 7-21 Keyword editing interface showing the new synonym for cruise control

4. Assign category to the keyword. Keywords must be associated with category to be used by the natural language processor. Follow these steps for category assignment:
 - a. From the Display Mode drop-down box, select **Category Tree**. See Figure 7-22.

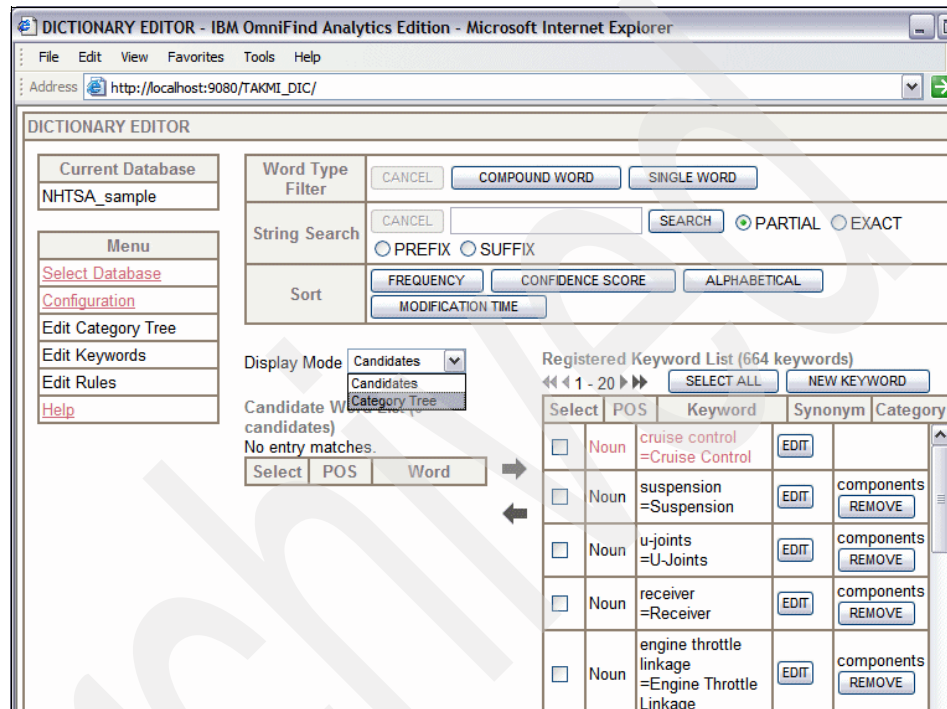


Figure 7-22 Choose Category Tree as display mode

The category tree will appear. See Figure 7-23.

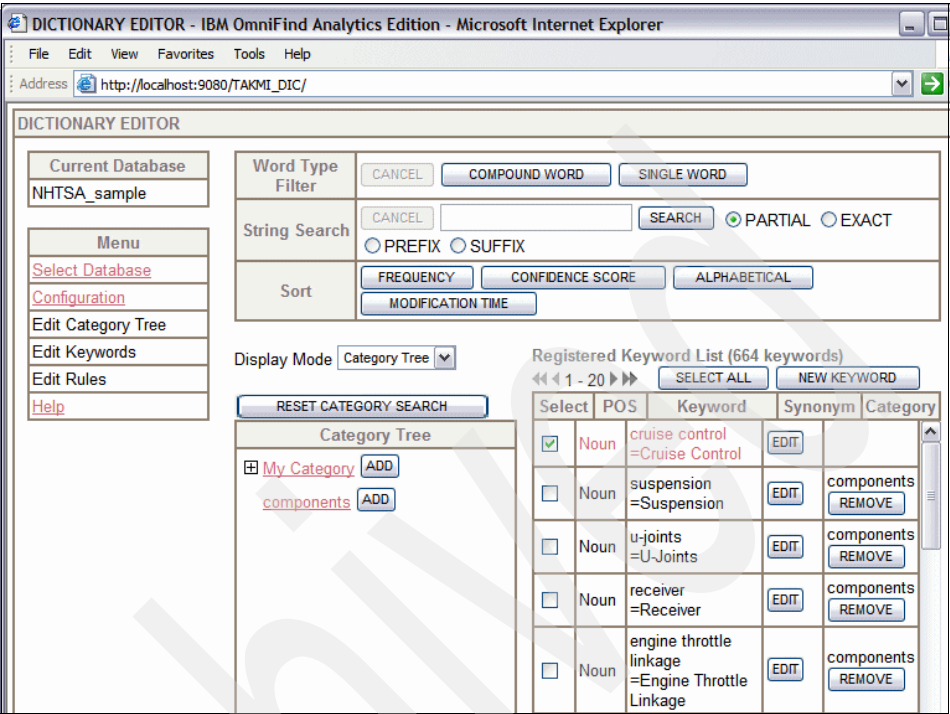


Figure 7-23 Assign keywords to category

- b. Select the check-box next to the keyword that you want to add to the category. For our example, we select the check-box next to cruise control. See Figure 7-23.
- c. Click **ADD** next to the category you want to associate the keyword to. For our example, we click **ADD** next to the components category.

The window will now display the category next to the keyword. See Figure 7-24.

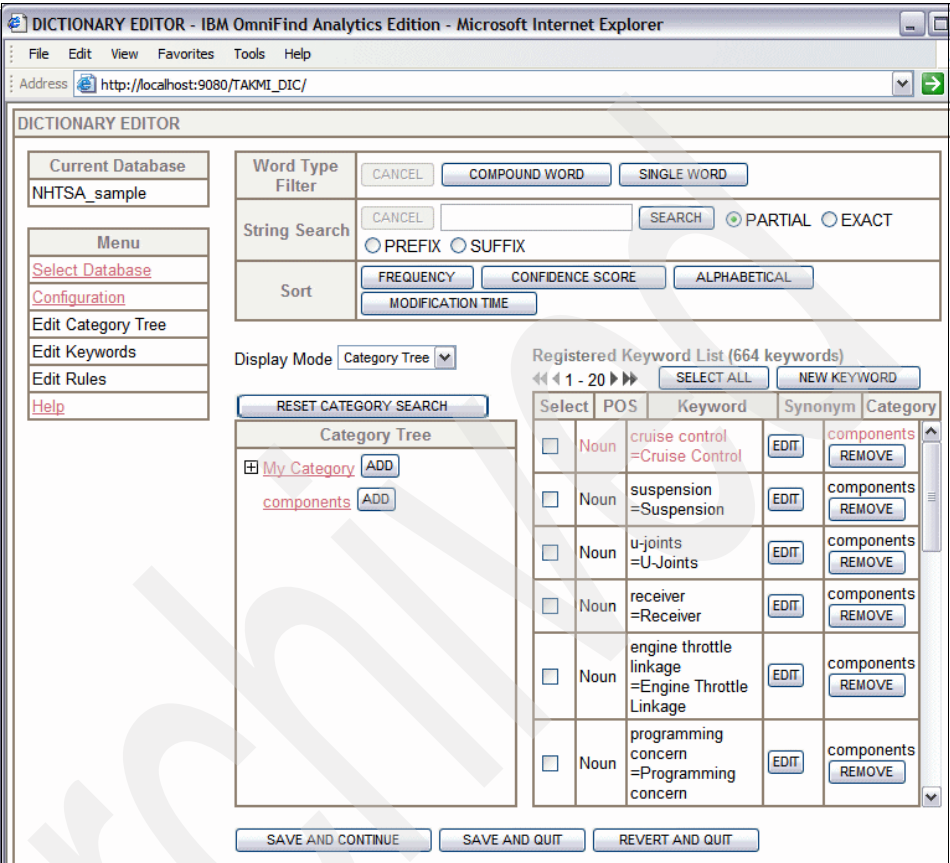


Figure 7-24 New keyword cruise control has been assigned with the components category

- Repeat the previous steps if you want to add more keywords, synonyms, and associate them with categories. When everything is done, click **SAVE AND QUIT**.

Searching for keywords

In a keyword file, there can be many keywords. For our example, we have 664 keywords. To create synonyms for a keyword or remove it or edit the keyword, you might have to search through the list to find the keyword. One way is to use the interface as shown in Figure 7-24 to find the keyword and then edit it or delete it. To search for the keyword:

1. Click **COMPOUND WORD** or **SINGLE WORD**. If you are searching for a compound word such as cruise control, click **COMPOUND WORD**.
2. Enter a search string.
3. Select **PARTIAL**, **EXACT**, **PREFIX**, or **SUFFIX**. If you enter a partial string for the keyword, select **PARTIAL**. If you enter the exact keyword string, you can select **EXACT**. If you know or enter only the prefix of the keyword, select **PREFIX**. If you know or enter only the suffix of the keyword, select **SUFFIX**.
4. Click **SEARCH**.

The system will find the keywords and display on the window. You can navigate through the keyword lists using the double-left arrow icon, left arrow icon, right arrow icon, and double-right arrow icon.

In addition, you can sort the keywords by frequency, confidence score, alphabetical, and modification time.

Removing keywords

From the window as shown in Figure 7-24, you can click **REMOVE** to remove the keyword from the list.

Customizing rules

Dictionary rules are one of the most powerful features in OmniFind Analytics Edition text mining. Rules allow the user to mine for virtually any concept expressible in natural language.

We cover the following topics:

- ▶ Dictionary rules overview
- ▶ The Rule Editor
- ▶ Regular expression syntax
- ▶ Creating a sample rule
- ▶ Adding additional constraints
- ▶ Summary of rule creation

8.1 Dictionary rules overview

Keywords are simply not enough, in terms of text mining. Although more powerful than simple string search, they are only individual words. Human beings can express a few concepts in single words, but most of our concepts require combinations of words.

The combinations are, again, more than just a series of keywords: Concepts are words linked in grammatical relations, sometimes quite complex relations. But, herein lies the meaning of the unstructured data. Rules give us the power to convert meaning and concepts into structured attributes.

Consider, this sentence, for example:

President Lincoln visited Gettysburg.

The basic text analytics in OmniFind Analytics Edition will tell us that President Lincoln is a proper noun, the visit is an action verb, and that Gettysburg is a place. But that is not the meat of what is being expressed here. The concept is that President Lincoln, a distinguished United States President, was in the town of Gettysburg. So, to get our analytics engine to answer the question, “Where has President Lincoln been?” we require a dictionary rule to establish a grammatical relation of President Lincoln and his location. Our rule can also help us find out what the President does, for example, visits battle sites, in this case. This is the tip of the iceberg: Although the Rule Editor makes the syntax of these rules manageable, a certain degree of linguistic skill is also necessary.

Every rule has the structure shown in Figure 8-1.



Figure 8-1 Anatomy of a rule

Here are all the constraint types, and a summary of what they mean:

- ▶ **String:** String can be entered here. You really should not use this for actual words, since the next constraint, Lex, was intended for words.
- ▶ **Lex:** This is short for Lexical item. The lexicon is what linguists use to refer to the dictionary we native speakers of a language keep in our brain. For our purposes, just think lex = word.
- ▶ **POS:** This stands for Part of Speech, used, when you are concerned with writing a rule that pertains to nouns, verbs, adjectives, adverbs, and others.
- ▶ **fters:** This stands for features. It is an OmniFind Analytics Edition specific notion, referring to more fine-grained grammar. ftrs include:
 - coordinating
 - subordinate
 - personal
 - wh
 - wh-possessive
 - there
 - genitive/possiveEnding
 - comparative
 - superlative
 - past
 - gerund/presentParticiple
 - pastParticiple
 - singular.3rd
 - 3rd.singular
 - aux
 - aux.past
 - aux.gerund/presentParticiple
 - aux.3rd
 - aux.modal
 - symbol
 - preposition
 - to.preposition
 - singular
 - plural
 - singular.proper
 - plural.proper
 - foreignWord.

5. **Category:** This is any category that you have configured.

Note: We cover each of these in detail. Do not worry if your grammar is rusty!

8.2 The Rule Editor

The Rule Editor is a J2EE Web application, bundled with OmniFind Analytics Edition, that aids the user in rule creation by providing a graphical interface. The rules that are created are associated with categories in the `category_tree.xml` configuration document. These rules are applied during natural language processing. The resulting annotation becomes part of the index, and the effects of the rules are seen in the Text Miner application.

The Rule Editor application can be accessed from a link in the Dictionary Editor application. See Figure 8-2.

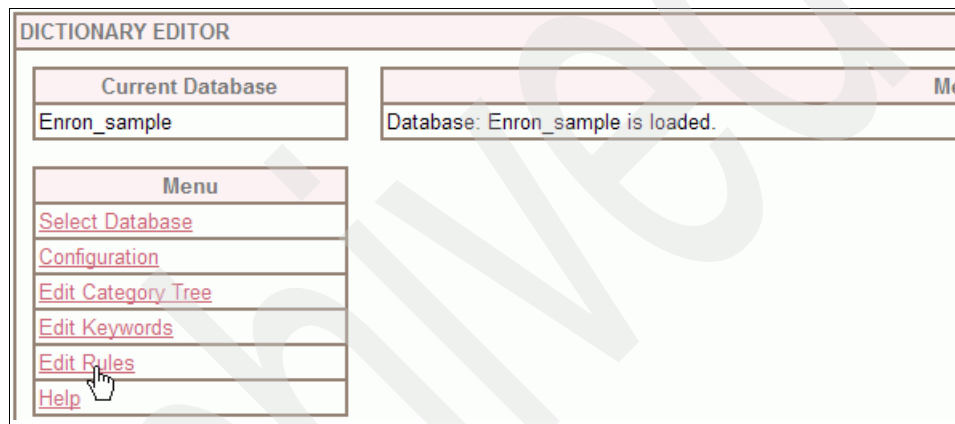


Figure 8-2 Launch the Rule Editor application via link

To create a new rule, click the **Add a Rule** link as shown in Figure 8-3.

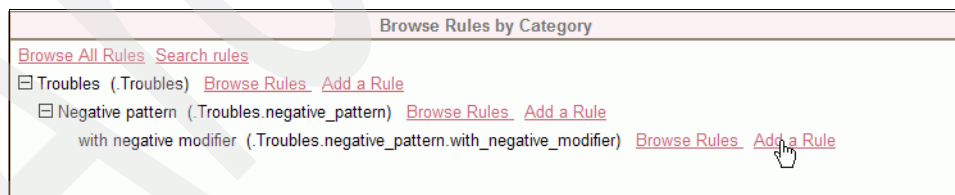


Figure 8-3 Add a rule link

It might be the case that you do not see the Add a Rule link. See Figure 8-4.

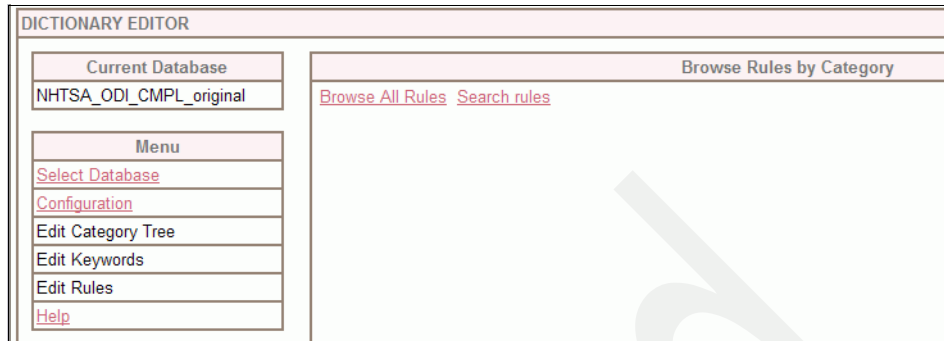


Figure 8-4 Why can't I add a rule?

This can only mean one thing: You have not configured any categories for this database. Rules must be associated with a category. You must create categories first before creating rules.

Note: If you have not created any categories yet, read Chapter 7, “Customizing the dictionary” on page 165. For the example in this chapter, we use the sample NHTSA database and assume a category structure of `.Troubles.negative_pattern.with_negative_modifier`.

8.3 Regular expression syntax

OmniFind Analytics Edition uses the Java `java.util.regex` package for regular expression matching. This is described in the Java regex reference (<http://java.sun.com/j2se/1.4.2/docs/api/java/util/regex/Pattern.html>). All rules with the `/` operator (see below) will be evaluated using `java.util.regex` Classes.

Note: In addition to the Java regex processing, all linguistic processing, including that done by LanguageWare®, will also apply.

The / operator

The `/` operator is *not part of the Java regex syntax*. It is used to tell OmniFind Analytics Edition's rule interpreter to invoke the Java regex processing on a rule. So, the following expression only matches with "love":

```
str="love"
```

However, the following expression matches with strings that contain "love" inside, such as "lover", "lovely", "beloved", "glove", and "loves" besides "love":

```
str="/love/"
```

Note: If the first "/" or the last "/" is missing, a constraint is *not* interpreted as regular expression: The constraint will be interpreted as string pattern matching.

The | (pipe) operator

The main thing we need for our rules is the | operator, called *pipe*. It is interpreted as the boolean *OR*. The pipe has the lowest precedence among all operators.

It is important, certainly when writing String constraints, to understand that the above constraint is a *literal* constraint — literal in contrast to a *regular expression*. This means that the rule will not be interpreted by the regular expression interpreter. In fact, it will not be interpreted at all: The only matches will be on exactly one of those words in the list. To understand the implications of this, let us review the regular expression syntax.

The ^ and \$ operators

^ matches only at the beginning of a line. \$ matches only at the end of a line. Here are some examples:

The following syntax matches with strings that start with "love" such as "lover", "lovely", and "loves", in addition to "love", but does not match "beloved" or "glove":

```
str="/^love/"
```

The following syntax matches with strings that end with "love" such as "glove" and "truelove", in addition to "love", but does not match "lovely" or "loves":

```
str="/love$/"
```

The following syntax matches with strings that start and end with "love". So it only matches with "love". In this case, you are back to square one; that is, you do not need to invoke the Java-regex interpreter if you really just want exactly that particular word to get matched:

```
str="/^love$/"
```

() Operators

If you want more than one word — and variations of those words — to be evaluated by the Java regex interpreter, you need to use parentheses. In short, () is the grouping operator, like any usual regular expression.

Consider the following syntax:

```
ab|cd
```

This is interpreted as:

`(a)(b|c)(d)`

And, it matches "abd" or "acd".

In contrast, consider this syntax:

`(ab)|(cd)`

It matches "ab" or "cd".

Here are a few concrete examples:

Consider this syntax:

`str="/love|hate/"`

This matches with any combination of the letter "l" + "ove" + "e" (the last letter of "hate"). Thus, the following expression would match:

`"lovee"`

"l" + "hate" would also work:

`"lhate"`.

Hmmm... not very useful in this case. If we want just either variations on the words "love" or "hate", we add parentheses around each of these words, like:

`str="/(love)|(hate)/"`

This has actually the same effect as:

`str="/((love)|(hate))/"`

It matches with "beloved", "truelover", "hater", "I love you", for example.

Remember: if you really only want the words "love" or "hate" and no variations, you do not need the Java regex, unless you use the regular expression:

`str="/^((love)|(hate))$/"`

This matches with either "love" or "hate". It would not match "I love you".

With this basic understanding of regular expression syntax, we can create our first rule, using the Rule Editor.

8.4 Creating a sample rule

Our first rule is going to capture a negative construction, such as:

designed poorly
working hardly
perform badly
smoking excessively

Note: Perhaps you know that adverbs can sometimes occur *before* the verb or *after* it: for example, *poorly* can always come either before or after the verb. *Hardly* is possible in both positions, but only marginally acceptable by native speakers post-positionally.

You also might know that adjectives can be formed using the participial forms of a verb: For example, the present participle of the verb *smoke* is *smoking*, and the past participle is *smoked*, which are used in phrases such as a *smoking engine* and *smoked salmon*.

This is a kind of expression that is negative, even though it does not contain the words of negation in English, such as “not” and “never.” In the end, our complete rule should look like this:

```
<w id="0" pos="/^((adjective)|(verb))$/"/>  
<w id="1" str="/^((excessively)|(poorly)|(badly)|(hardly))$/"/>
```

This rule translates to “find any combination of (any) adjective *or* (any) verb in the first (0) position, followed by any one (but not all) of the strings “excessively” or “poorly” or “badly” or “hardly”, in the second (1) position.

Note: The “w” positions start numbering from 0, not 1, just like arrays in Java.

We also need another rule which handles the cases where the adjective or verb comes in the second position, as in:

poorly functioning
hardly runs
badly smoking
excessively loud

This rule is simply the opposite of the first one:

```
<w id="0" str="/^((excessively)|(poorly)|(badly)|(hardly))$/"/>  
<w id="1" pos="/^((adjective)|(verb))$/"/>
```

We demonstrate only the first of these in this chapter. We now create a sample rule for the `.Troubles.negative_pattern.with_negative_modifier` category.

To create the sample rule, we provide the instructions for the following tasks:

1. Creating a POS constraint
2. Creating a string constraint
3. Setting the rule name and its value
4. Manually editing the rule for additional rule constraints
5. Converting the rule file from `.rpf` to `.pat`
6. Testing rules

8.4.1 Creating a POS constraint

First, we have to create a POS constraint for the sample rule:

```
<w id="1" pos="/^((adjective)|(verb))$/"/>
```

To create the POS constraint as shown above, follow these steps:

1. Launch the Dictionary Editor if you have not done so. Select the database that you want to work with. Select **Edit Rules**.
2. Select **Add a Rule** (Figure 8-3 on page 192) next to the category.
3. You will now see that you have to *Select constraint type*. Select **POS** for your constraint type (see Figure 8-5).

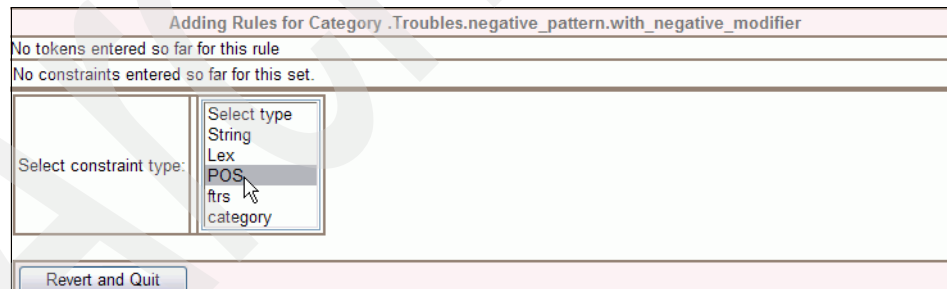


Figure 8-5 Add a POS constraint

4. Using the *Select an operator* box on the right, select the **left parenthesis** (see Figure 8-6).



Figure 8-6 Select an operator.

- Continue building your rule by selecting the next left parenthesis, and then select **adjective** from the *Select value* box on the left. It should ultimately look like what is shown in Figure 8-7.

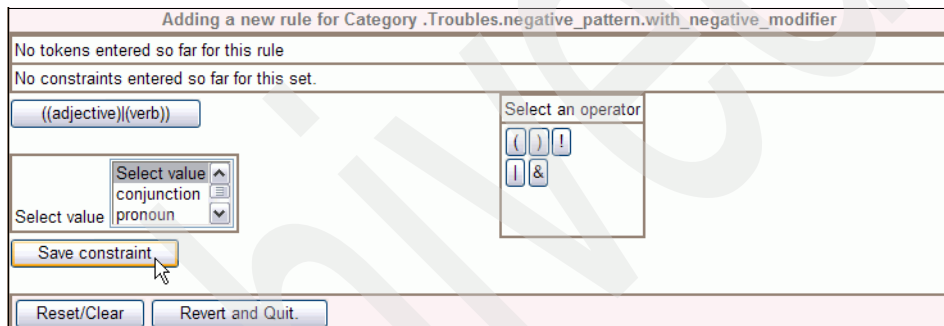


Figure 8-7 POS constraint ((adjective)|(verb))

- After completing the POS constraint, click **Save constraint**.
- Our rule has a second constraint, or, *token*, namely, the string constraint. Before proceeding to the next section, click **Save current constraints and Add next token**. See Figure 8-8.

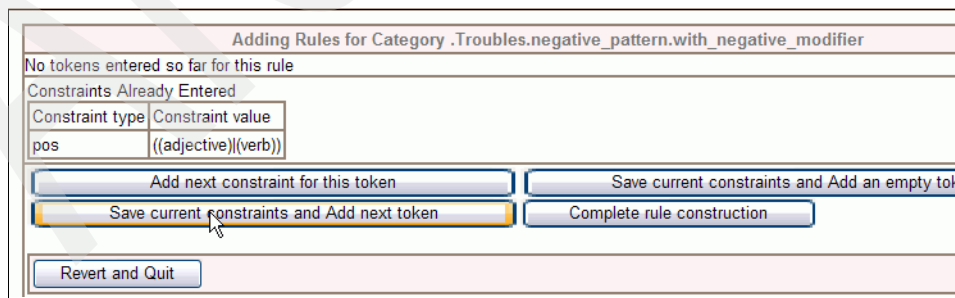


Figure 8-8 Save the first token in the rule

We have just created the following POS constraint:

```
((adjective)|(verb))
```

What we needed for our sample rule is:

```
/^((adjective)|(verb))$/
```

We will go back to this later to show you how to complete the rest of the constraint.

8.4.2 Creating a string constraint

As we mentioned above, string rules can be any combination of ASCII characters. The easiest example would be if you had a rule looking for e-mail addresses, which would be found by a pattern like String @ String, because the commercial ‘at’ sign has no grammatical function: it is really just a string.

We are looking for the words *excessively*, *poorly*, *badly*, and *hardly*. It is clear that these would not be matched just using POS. They are not just any adverbs — they must be exactly *these* adverbs, which have a negative connotation to them. So, we will match them as strings.

Continuing from the previous section, we want to create a string constraint as follows:

```
<w id="0" str="/^((excessively)|(poorly)|(badly)|(hardly))$/"/>
```

To create the string constraint for our example, follow these steps:

1. Select **String** as shown in Figure 8-9.

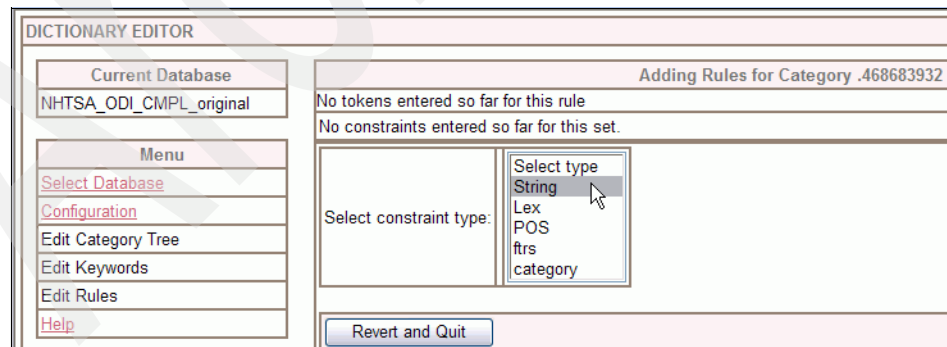


Figure 8-9 Select String for your rule's constraint

You are immediately directed to the next window. If you understood the discussion on regular expressions (see 8.3, “Regular expression syntax” on page 193), you know that our original expression would be a miserable failure:

```
str="excessively|poorly|badly|hardly"
```

Because it has no / operator, it will be evaluated as a “pure” string. That is, the pipe | operators will be interpreted as ASCII characters, *not* OR. Thus, the constraint above would only match the *exact* text, *excessively|poorly|badly|hardly*. We need to construct the regular expression:

```
str="/^((excessively)|(poorly)|(badly)|(hardly))$/"
```

This will only match strings that *begin* with any of the *exact* words in the rule. The reasoning is as follows: We use ^ and \$ because we do not want any variations in the words *excessively*, *poorly*, *badly* or *hardly*. This will invoke the Java regex interpreter, and only match expressions which have exactly one of the words in parentheses. This is similar to the last example in our discussion of regular expression syntax in the foregoing discussion (see 8.3, “Regular expression syntax” on page 193).

- 2. Select the (operator *twice*, enter *excessively*, above, and then close the parentheses. See Figure 8-10.

Adding a new rule for Category .Troubles.negative_pattern.with_negative_modifier

Tokens entered so far

pos=((adjective))(verb))

No constraints entered so far for this set.

⌕

Enter value

excessively

Submit

Select an operator

(

)

!

|

&

Save constraint

Figure 8-10 Inserting operators

- 3. Insert the pipe | operator, and repeat the prior process for the other three words. Do not forget to close parentheses around each word, and then close the entire constraint with one parenthesis at the very end. See Figure 8-11.

200

Introducing OmniFind Analytics Edition: Customizing for Text Analytics

Adding a new rule for Category .Troubles.negative_pattern.with_negative_modifier

Tokens entered so far

pos=((adjective))((verb))

No constraints entered so far for this set.

Enter value

Select an operator

Figure 8-11 Complete str constraint

4. Click **Save constraint**.
5. We should now have two constraints in our rule. Figure 8-12 shows what our constraint looks like. Click **Complete rule construction**.

Tokens entered so far

pos=((adjective))((verb))

Constraints Already Entered

Constraint type	Constraint value
str	((excessively))((poorly))((badly))((hardly))

Figure 8-12 Complete this rule's construction

8.4.3 Setting the rule name and its value

There are several things we *can* do to complete the rule, but only two things we *must* do: we must give the rule a *name*, and give the rule a *value*. This name is internal to the tool; the value is what you will see in the Text Miner user interface. Fill in an easily understandable name. The value requires some explanation.

If you entered the rule value as a string, you would actually see that string, for example, "negative pattern," displayed as your search results. Instead, you would probably rather see the phrases that match your rule. For that, we have to use the *Enter rule value interactively* feature. It means that the displayed values will be the search results returned by applying the rule.

To complete the rule creation, follow these steps:

1. Enter a name for the rule.
2. Select the **Enter rule value interactively** radio button.
3. Choose **0** from the Select variable drop-down box as shown in Figure 8-13.

Rule name:

Please check the appropriate radio button to select the rule value format.

☐ Rule value (as string):

☒ Enter rule value interactively

Rule value:

Figure 8-13 Displays the value of the variable at position 0

4. Because many features are returned as values of the search result, the Select feature function allows you to choose which feature to display. We want to see the actual words that occur in Position 0 — in our constraint, remember, these are either adjectives or adverbs. To achieve this, select **lex** for the feature. See Figure 8-14.

Rule name:

Please check the appropriate radio button to select the rule value format.

☐ Rule value (as string):

☒ Enter rule value interactively

Rule value:

Figure 8-14 The lex feature displays the actual words returned by the constraint

5. Click **Add rule value**. This will set the value for the first constraint. See Figure 8-15.

Rule name:

Please check the appropriate radio button to select the rule value format.

☐ Rule value (as string):

☒ Enter rule value interactively

Rule value: Select variable Select feature

Figure 8-15 Set the display value for the first constraint

6. Repeat this process for the second constraint — indicate that the value is for the second constraint by setting the variable to **1**. Make sure to click **Add rule value** again when you are done.

The complete values should look like this:

`$(0.lex) ${1.lex}`

See Figure 8-16.

Rule name:

Please check the appropriate radio button to select the rule value format.

☐ Rule value (as string):

☒ Enter rule value interactively

Rule value: Select variable Select feature

Figure 8-16 The last step in our first rule!

7. It would also be possible here to alter the number of occurrences of the tokens, or, change their ordering within the rule. For now, we will not. Click **Save rule** and the Rule Editor will exit.

Note: Some browsers will ask to confirm the fact that the application is attempting to close the browser window.

8.4.4 Manually editing the rule for additional rule constraints

Our rule is not done yet — before we can implement the rule, we have to insert the / , ^, and \$. operators.

Note: As of this writing, OmniFind Analytics Edition Rule Editor does not support the addition of diacritic characters such as / , ^, or \$. These must be added to the configuration manually.

The Rule Editor creates a file in the <database>/pattern directory (see Figure 8-17), with the extension .rpf, which stands for *Rich Pattern Format*.

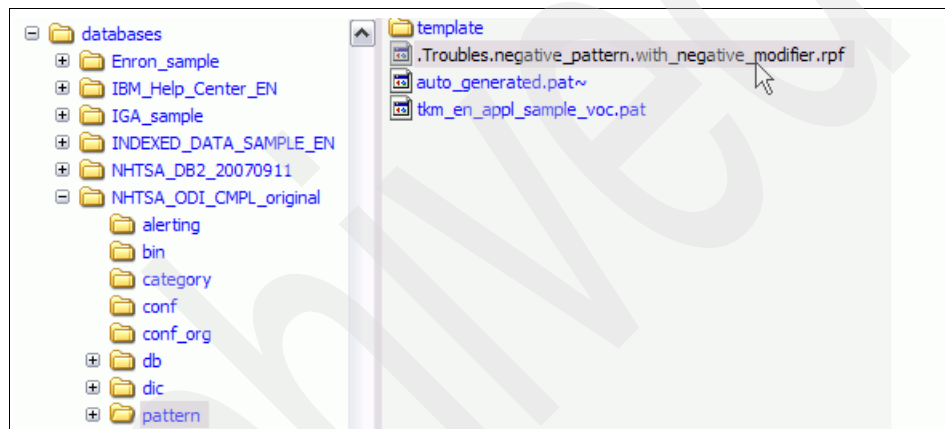


Figure 8-17 The .rpf file in the patterns directory

Note: The .rpf file name will be the same as the category name, found in category-tree.xml.

Open the .rpf file, and edit it so that it looks like what is shown in Example 8-1. The bold text contains the lines that you need to edit manually.

Example 8-1 Manually editing the rule

```
<?xml version="1.0" encoding="UTF-8"?>
<pattern-list lang="eng">
<mi
category=".Troubles.negative_pattern.with_negative_modifi
er"
name="negative_pattern_post_position" value=" ${0.lex} ${1.lex}">
<b>w id="0" optional="{1,1}" pos="/^((adjective)|(verb))
```

```

$/"/>
<w id="1" optional="{1,1}"
str="/^((excessively)|(poorly)|(badly)|(hardly))$/"/>
</mi>
</pattern-list>

```

Now, it is time to test our rule.

8.4.5 Converting the rule file from .rpf to .pat

Up until Version 8.4 of OmniFind Analytics Edition, the rules were saved in .pat files, which used the TAKMI format. As of Version 8.4, rules are saved in .rpf. However, OmniFind Analytics Edition will only be aware of rules in .pat files. Hence, *the .rpf files must be converted to .pat format.*

Note: The only difference between TAKMI format is that rules in .rpf files have an additional (optional) attribute for each word tag ("w"). The attribute takes values of the form {i,j}, where both i and j are integers. The semantics of the optional attribute are that the pattern should match i to j repetitions of the corresponding word. There is one other small difference you might notice: the .pat file does not give the rule's *name*, just the *category* and the *value*.

Example 8-2 shows the rules in the .rpf file.

Example 8-2 Sample rule in .rpf file

```

<?xml version="1.0" encoding="UTF-8"?>
<pattern-list lang="eng">
<mi
category=".Troubles.negative_pattern.with_negative_modifier"
name="negative_pattern_post_position" value="{0,1} {1,1}">
<w id="0" optional="{1,1}" pos="/^((adjective)|(verb))
$/"/>
<w id="1" optional="{1,1}"
str="/^((excessively)|(poorly)|(badly)|(hardly))$/"/>
</mi>
</pattern-list>

```

Example 8-3 shows the rules in .pat format. The differences between both files are the additional bold text in Figure 8-3.

Example 8-3 Sample rule in .pat file

```
<?xml version="1.0" encoding="UTF-8"?><pattern-list lang="eng">
<mi label="-0"
category=".Troubles.negative_pattern.with_negative_modifier" value="
${0.lex} ${1.lex}">
<w id="0" pos="/^((adjective)|(verb))$/"/>
<w id="1" str="/^((excessively)|(poorly)|(badly)|(hardly))$/"/></mi>
</pattern-list>
```

To convert .rpf file to a .pat file, use the command:

takmi_transform_rules.bat/sh

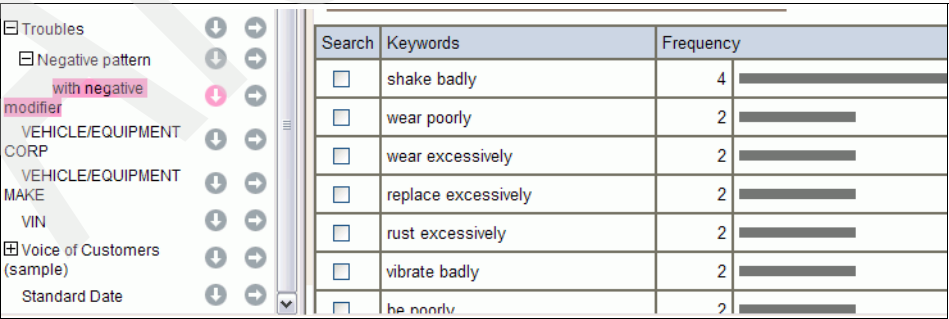
This is in the <OAE_HOME>/bin directory. The syntax and execution of this command is as follows:

takmi_transform_rules.bat/sh <source .rpf file> <destination .pat file>

Note: The generated .pat file must be in the <database>/pattern directory before NLP processing. The .rpf file is not recognized at the time of this writing.

8.4.6 Testing rules

To test rules you have written, you must allow the Natural Language Processor to annotate the .atml. This means that you run the takmi_nlp.bat/sh command on your database. Of course, a new index must be generated, as well. (if you are not familiar with the steps in this process, see 4.6, “Natural language processing” on page 78). Minimally, you will need to reload the database, which might mean restarting the Text Miner application. If you have done everything correctly, you will see that your rule has been applied. See Figure 8-18.



Search	Keywords	Frequency
<input type="checkbox"/>	shake badly	4
<input type="checkbox"/>	wear poorly	2
<input type="checkbox"/>	wear excessively	2
<input type="checkbox"/>	replace excessively	2
<input type="checkbox"/>	rust excessively	2
<input type="checkbox"/>	vibrate badly	2
<input type="checkbox"/>	be poorly	2

Figure 8-18 Rule applied

In a future release of OmniFind Analytics Edition, there will be an NLP browser user interface, which will confirm the application of a rule. Figure 8-19 shows a preview.

DB: NHTSA_ODI_CMPL_original Result Type: Keyword TAE initialization ☐

replace excessively

Process(POST)

id	range	category	value
3	8-19	Adverb	excessively
1	0-19	with negative modifier	replace excessively
2	0-7	Verb	replace

Figure 8-19 Technology preview - Verification of rule's application

8.5 Adding additional constraints

In this section, we show you how to perform the following tasks:

- ▶ Adding a lex constraint
- ▶ Adding a category constraint
- ▶ Adding an ftrs constraint

8.5.1 Adding a lex constraint

The main function of using a lex constraint is that, in addition to invoking the Java regex processor, OmniFind Analytics Edition's own natural language processing parsers also perform grammar analysis. Specifically, the process of lemmatization is performed. It is worth the time to describe lemmatization, and compare it to a less sophisticated technology, called stemming (refer to <http://publib.boulder.ibm.com/infocenter/discover/v8r4/index.jsp?topic=/com.ibm.discovery.es.common.doc/standard/iisgloss.htm>).

Stemming uses an algorithm to determine the *stem* of a word. Stemming is a form of linguistic normalization that reduces a set of words to the lowest common denominator. For example, the words *connections*, *connective*, and *connected* are reduced to *connect*. While this is intuitive for many words, it has limitations when words have a similar base form, but are not related in meaning. In stemming, the stem does not need to be the same as the morphological root of the word. This has been a problem in natural language processing since 1968¹.

¹ Lovins, J. B. "Development of a Stemming Algorithm." Mechanical Translation and Computational Linguistics 11, 1968, 22--31.

For example, while stemming correctly links *run* to *running*, it will never associate *ran* to either of these, even though it is a form of the same lexical item. Also, there is the possibility of false positives, such as *Maryland* -> *Mary*, or *land* -> *marigold*. Lemmatization differs from stemming in that stemming is *algorithmic* and generally *does not operate with a dictionary* that lists the words of a language. Lemmatization requires a dictionary for both indexing and searching.

Lemmatization, used in OmniFind Analytics Editions, is a form of linguistic processing that determines the *lemma* for each word form that occurs in text. The lemma of a word encompasses its base form plus inflected forms that share the same *part of speech*. For example, the lemma for *go* encompasses *go*, *goes*, *went*, *gone*, and *going*. Lemmas for nouns group singular and plural forms (such as *calf* and *calves*). Lemmas for adjectives group comparative and superlative forms (such as *good*, *better*, and *best*). Lemmas for pronouns group different grammatical *cases* of the same pronoun (such as *I*, *me*, *my*, and *mine*).

We can illustrate this with our earlier string rule involving *hardly*. As you know, *hardly* is an adverb meaning barely, or almost not. However, *hardly* has nothing to do with the adjective *hard*, meaning difficult. Stemming would link the two words to the stem *hard*, which might produce anomalies such as:

hard working --> hardly working

Because *hardly* was listed in our earlier rule as part of a string constraint, the lemmatization is not applied. In our next rule, we will list *hard* in a *lex* constraint, which will cause lemmatization to apply. This will have the effect of first registering the part of speech of *hard*, which is adjective. Second, it will link *hard* to its comparative and superlative forms *harder* and *hardest*. It will *not* link *hard* to *hardly* because the two have different parts of speech, and are different lexical (dictionary) entities. We will use the Rule Editor to create the following rule:

```
<w id="0" lex="/^((hard)|(difficult))$/"/>
  <w id="1" lex="to"/>
  <w id="2" pos="verb"/>
```

The second w-node is also worth a brief look. We are trying to pick out all the phrases like:

hard to brake
hard to start

These are of the form

```
<w id="0" lex="/^((hard)|(difficult))$/"/>
  <w id="2" pos="verb"/>
```


That is, “to brake” is the infinitive form of the verb brake. However, that rule would also pick out

hard working

This is (usually) not considered a trouble, which is what we are looking for. So, one way to get around this is to add *to* as a lex constraint.

The process for using the Rule Editor to create this rule is the same as with the other rules, so we will not go into as much detail. The main difference is the choice of Lex as a constraint type. See Figure 8-20.

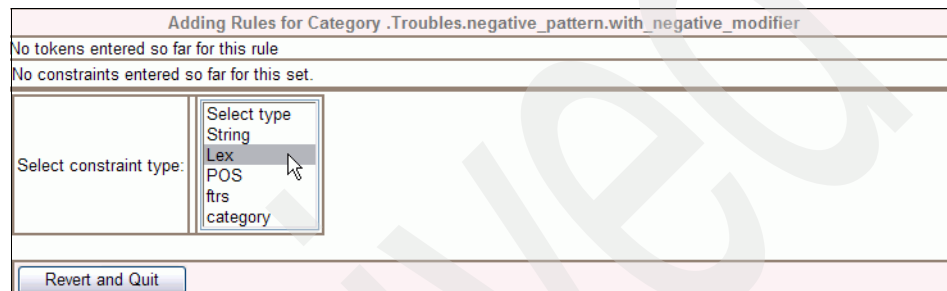


Figure 8-20 Creating a Lex constraint

For completeness' sake, we will also add the word *difficult*, which, in this constraint, functions as a synonym for *hard*. See Figure 8-21.

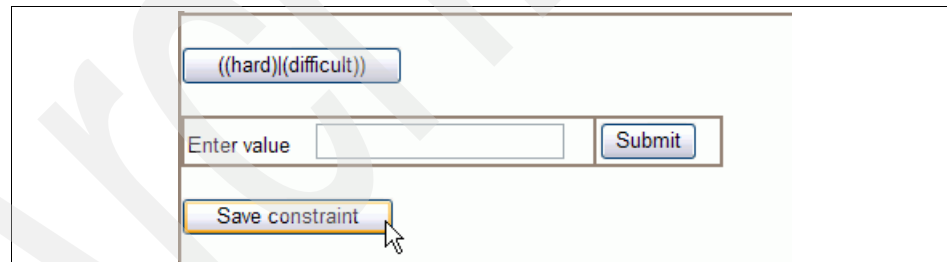


Figure 8-21 hard and difficult as synonyms

Continue as before to save this constraint, and add the next, which is *to* as a Lex constraint. Finally, add the POS constraint for a Verb. Your rule should look like Figure 8-22.

Token	Repeats?	Reorder	Delete token
lex=((hard))((difficult))	Min: <input type="text" value="1"/> Max: <input type="text" value="1"/> <input type="button" value="Update"/>		<input type="button" value="Delete"/>
lex=to	Min: <input type="text" value="1"/> Max: <input type="text" value="1"/> <input type="button" value="Update"/>	<input type="button" value="Move up"/>	<input type="button" value="Delete"/>
pos=verb	Min: <input type="text" value="1"/> Max: <input type="text" value="1"/> <input type="button" value="Update"/>	<input type="button" value="Move up"/>	<input type="button" value="Delete"/>

Rule name:

Please check the appropriate radio button to select the rule value format.

☐ Rule value (as string):

☒ Enter rule value interactively

Rule value:

Select variable

Select feature

Figure 8-22 Final rule.

Notice that we have set the rule values, as before, to the resulting lex values that get returned for positions 0,1,2. After saving the rule, make sure to open the .rpf file in an editor, and enter the regex diacritics into the first lex element:

```
<w id="0" lex="/^((hard)| (difficult))$/" optional="{1,1}"/>
```

Finally, generate the .pat version. Because this rule is part of the same category, there is no need to create a new .pat file. Execute the command exactly as you did before, and the entry will get appended to the original .pat file. Additionally, you will need to run NLP preprocessing as before. If everything is correct, you will see that your category has expanded to include the new Lex rule (Figure 8-23).

[-] Troubles	↓	→	<input type="checkbox"/>	vibrate badly	2	██████
[-] Negative pattern	↓	→	<input type="checkbox"/>	difficult to explain	2	██████
with negative modifier	↓	→	<input type="checkbox"/>	rust excessively	2	██████
VEHICLE/EQUIPMENT	↓	→	<input type="checkbox"/>	difficult to operate	2	██████
CORP	↓	→	<input type="checkbox"/>	hard to open	2	██████
VEHICLE/EQUIPMENT	↓	→	<input type="checkbox"/>	be poorly	2	██████
MAKE	↓	→	<input type="checkbox"/>	hard to believe	2	██████
VIN	↓	→	<input type="checkbox"/>	difficult to tighten	2	██████
[-] Voice of Customers (sample)	↓	→				
Standard Date	↓	→				

Figure 8-23 New rule with “hard to” now functioning

8.5.2 Adding a category constraint

For a category constraint example, we will be using the Lotus® Notes® E-mail database of the correspondence between the team members that created this book, which was ingested through OmniFind Enterprise Edition. We have created a category structure of .names.person.fname and .names.person.lname to hold first and last names found in the dataset. For these categories, we configured dictionary entries (not rules²) with lists of first names and last names (and Internet ID's, although these are not relevant to the discussion here). Complete instructions for creating keyword (.adic) files can be found in Chapter 7, “Customizing the dictionary” on page 165. Example 8-4 shows the partial names.adic file.

Example 8-4 Partial names.adic file

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
  <entry id="0" lex="Ratheesh" pos="noun" cat=".names.person.fname"
mtime="1193016789016"/>
  <entry id="1" lex="Joel" pos="noun" cat=".names.person.fname"
mtime="1193016789031"/>
  <entry id="2" lex="Yuta" pos="noun" cat=".names.person.fname"
mtime="1193016789031"/>
  <entry id="3" lex="Harms" pos="noun" cat=".names.person.lname"
mtime="1193016583734"/>
  <entry id="4" lex="balunaini" pos="noun"
cat=".names.person.inet_email_id" mtime="1193016710188"/>
  <entry id="5" lex="polm" pos="noun"
cat=".names.person.inet_email_id" mtime="1193016710219"/>
```

² Rules cannot match with categories for which other rules have been assigned. The “category” attribute specifies a constraint by the category path assigned by the dictionary lookup annotator.

```
<entry id="6" lex="Srinivas" pos="noun" cat=".names.person.fname"
mtime="1193016825984"/>
<entryC id="7" str="Varma" pos="noun" eid="6"/>
<entryC id="8" str="&quot;Varma&quot;" pos="noun" eid="6"/>
<entry id="9" lex="Zhu" pos="noun" cat=".names.person.lname"
mtime="1193016583781"/>
<entry id="10" lex="Mobley" pos="noun" cat=".names.person.lname"
mtime="1193016583766"/>
<entry id="11" lex="ratheeshm" pos="noun"
cat=".names.person.inet_email_id" mtime="1193016710156"/>
<entry id="12" lex="Leslie" pos="noun" cat=".names.person.fname"
mtime="1193016789141"/>
```

It has always been a common practice in computing to put together the first and last names from a dataset. Using these two categories as constraint values, we can create a rule for this. We have also set up a category called “First and Last” to associate with the new rule.

So, in the Rule Editor, to add this category constraint, follow these steps:

- 1. Select **Add a Rule** next to First and Last (see Figure 8-24).



Figure 8-24 Adding the First Name Last Name rule

- 2. Select category as the constraint type (see Figure 8-25).

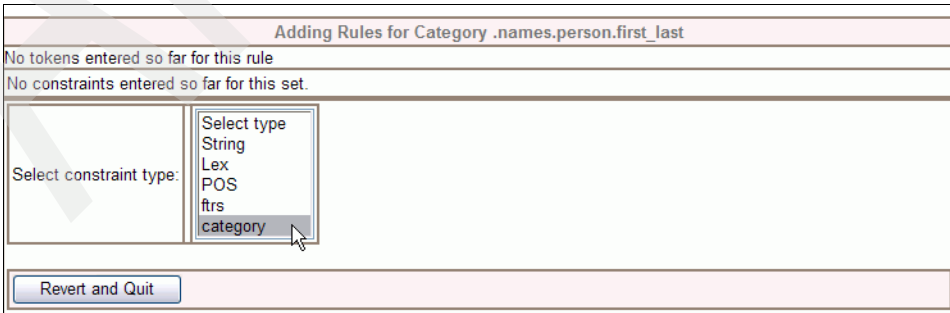


Figure 8-25 The category constraint

3. In the scroll bar menu, you can see the First Name and Last name categories. Choose **First name** for the first token (Figure 8-26).

Figure 8-26 First Name category is the first token

4. As before, click **Save constraint**, and the **Save all constraints and Add next token** button. Continue on with Last Name as the category for your second token. Clicking **Finish rule construction** brings you to Figure 8-27.

Token	Repeats?	Reorder	Delete token
category=First Name	Min: 1 Max: 1 Update		Delete
category=Last Name	Min: 1 Max: 1 Update	Move up	Delete

Rule name:

Please check the appropriate radio button to select the rule value format.

☐ Rule value (as string): Add rule value

☒ Enter rule value interactively

Rule value: 1 lex Add rule value

Save rule

Figure 8-27 The Finish rule window

5. As before, give the rule a name. For the name value, we enter this interactively, setting the display to be, just as the end user probably expects, the First Name followed by the Last Name. Save the rule.

At the time of this writing, you have to edit the .rpf file that gets created. This is because the category attribute in a rule must follow the category path which is assigned by dictionary lookup annotator, but the Rule Editor enters the category name. Example 8-5 shows the edited .rpf file. The bold text shows where you have to make the modification.

Example 8-5 Editing the .rpf file

```
<mi label="-0" category=".names.person.first_last" value=" ${0.lex}
${1.lex}">
<w id="0" category="First Name"/>
<w id="1" category="Last Name"/>
</mi>
```

You must change these values to the category paths:

```
<mi label="-0" category=".names.person.first_last" value=" ${0.lex}
${1.lex}">
<w id="0" category=".names.person.fname"/>
<w id="1" category=".names.person.lname"/>
</mi>
```

6. Convert the .rpf file into .pat format, run NLP processing, and rebuild your index. See 8.4.5, “Converting the rule file from .rpf to .pat” on page 205 as reference.

If everything is correct, you will see that the category rule combines the first names with the correct last names shown.

8.5.3 Adding an ftrs constraint

Features (ftrs) require some in-depth knowledge of grammar, beyond parts of speech. It is beyond the scope of this book to explain what each of the features can do. We will focus on a single example that illuminates as to the overall power of feature constraints in OmniFind Analytics Edition.

The sample ftrs rule we will write is fairly common in e-mail databases: there is almost always the need to pick out e-mail addresses. While there are a number of ways to get OmniFind Analytics Edition to do this, one of the simplest is to write a rule as shown in Example 8-6.

Example 8-6 E-mail address rule

```
<?xml version="1.0" encoding="UTF-8"?>
<pattern-list lang="eng">
<mi category=".addresses.email.internet_email_address"
name="" value=" $Internet E-mail address">
```

```
<w ftrs="symbol" id="0" str="@"/>
</mi>
</pattern-list>
```

We use an e-mail database of the correspondence between the team members that created this book, which was ingested through OmniFind Enterprise Edition and category .addresses.email.internet_email_address to accommodate our ftrs rule.

To create the ftrs rule:

- 1. From the Rule Editor, select **Add a rule** next to the Internet category (Figure 8-28).



Figure 8-28 Always begin by clicking Add a Rule

- 2. Select **ftrs** as your constraint type (Figure 8-29).



Figure 8-29 Select ftrs for constraint type

- 3. The OmniFind Analytics Edition natural Language Processor automatically annotates symbols for the given language. This is how the feature (ftrs) symbol is enabled. On the next window (Figure 8-30), select the symbol as your constraint, and click **Save constraint**.

Adding a new rule for Category .addresses.email.internet_email_address

No tokens entered so far for this rule

No constraints entered so far for this set.

symbol

superlative
wh
symbol

Select fr/s

Save constraint

Select an operator

() !
| &

Figure 8-30 Add symbol as the first constraint value

- This time, instead of adding another token, we extend this token with a further constraint. We have told the NLP engine that we want a symbol, but we have not told it which symbol. This is too broad — not a sufficient constraint, because we would get all strings containing any symbols. Simply enough, we add the string @, for the commercial ‘at’ symbol. Click **Add next constraint for this token** (Figure 8-31).

Adding Rules for Category .addresses.email.internet_email_address

1 tokens entered so far for this rule

Constraints Already Entered

Constraint type	Constraint value
frs	symbol

Add next constraint for this token

Save current constraints and Add an empty token

Save current constraints and Add next token

Complete rule construction

Figure 8-31 Add an additional constraint for a single token

- Notice on the next window that you are restricted in your choices of constraint values. You are not able to add another frs constraint. Select String for your constraint, and type in the @ (Figure 8-32).

Adding a new rule for Category .addresses.email.internet_email_address

No tokens entered so far for this rule

Constraints Already Entered

Constraint Type	Constraint Value
frs	symbol

Enter value @

Submit

Save constraint

Select an operator

() !
| &

Figure 8-32 Add the string @ as the value for your second constraint

6. Click **Submit**, and then go on to Save constraint and Complete rule construction, as in all the other examples.
7. Give a meaningful name to the rule.
8. Set the rule value.

This is a good rule to illustrate why you might want to enter the Rule value as a string. You only have a single token in this rule, and that token will always contain only one value: the @ symbol. So, your only option for displaying a value would be to display the @ in your Text Miner category page. it would look like Figure 8-33.

The screenshot shows the 'Category' tab in the Text Miner interface. The category is 'Internet'. The 'List' section has 'Keywords' selected. The 'Sort' section has 'Frequency' selected. The 'Max lines' section has '100' selected. The 'AND OR NOT' section has 'AND' selected. The 'Search' section shows a table with columns 'Keywords', 'Frequency', and 'Correlation'. The table has one row with the value '@' in the 'Keywords' column, '158' in the 'Frequency' column, and '1.0' in the 'Correlation' column.

Search	Keywords	Frequency	Correlation
<input type="checkbox"/>	@	158	1.0

Figure 8-33 Not so intuitive to display the $\{0.lex\}$ value

While the result set is what you are really interested in, the message to the user might be clearer if you imply that the returned results are Internet E-mail addresses. Enter Internet E mail address in the Rule value (as string) field (Figure 8-34). Click **Add rule value** after entering the String value.

The screenshot shows the 'Rule editor' in the Text Miner interface. The 'Token' field is 'ftsr=symbol' and the 'Repeats?' field is 'str=@'. The 'Rule name' field is 'inet_email'. The 'Rule value (as string)' field is 'Internet E-mail address'. The 'Rule value' field is 'Select variable' and the 'Select feature' field is 'Select'. The 'Add rule value' button is highlighted.

Rule name:

Please check the appropriate radio button to select the rule value format.

☒ Rule value (as string):

☐ Enter rule value interactively

Rule value:

Figure 8-34 Add rule value as a string

9. Save the rule, run the NLP and indexing as before, and test.

You will see that all Internet e-mail addresses are returned, under the category “Internet E-mail Address”, and that the results are the same (number) as if you had `{0.lex}` as a value (Figure 8-35).

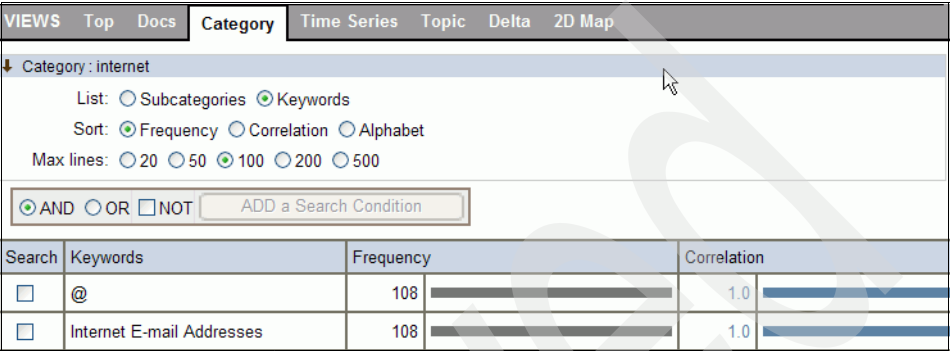


Figure 8-35 Choose how you want your rule to display

Finally, go to the Docs tab to confirm.

8.6 Summary of rule creation

Figure 8-36 provides a flow chart view of the rule creation process that you can refer to.

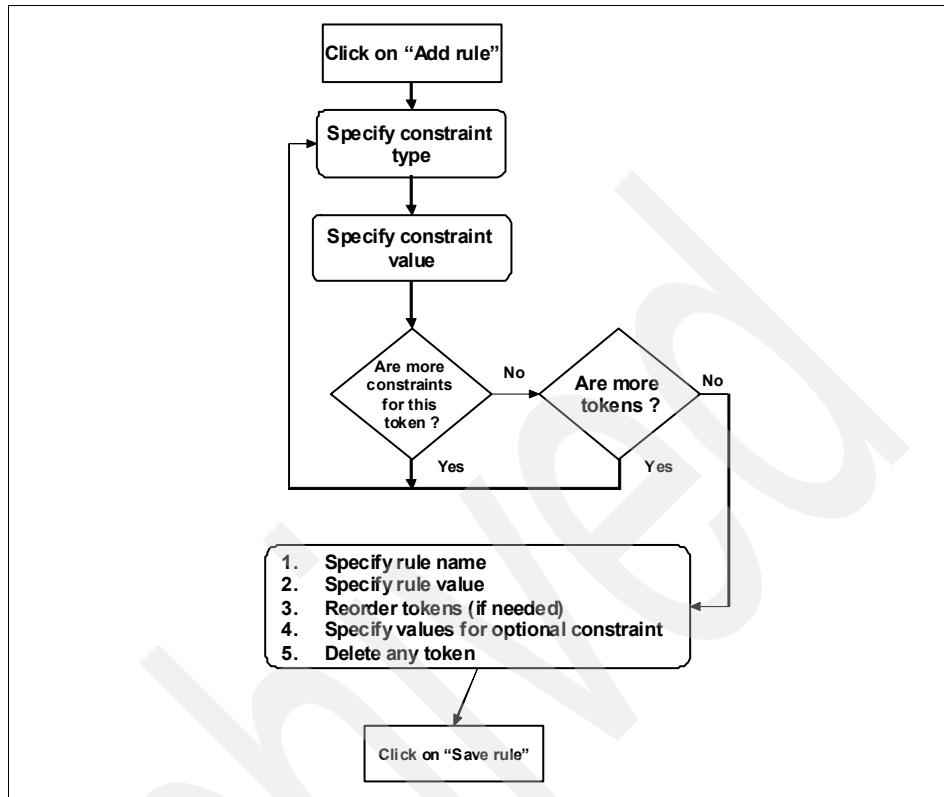


Figure 8-36 Flow chart for rule creation

Here are the additional steps you need to perform to test the rules you create:

1. Add any regex diacritic makers to the .rpf file created by the Rule Editor.
2. Convert .rpf file to .pat file format.
3. Run Natural Language Processing on the database.

If your database was created using OmniFind Enterprise Edition, you need to do a full re-crawl of the data. Then, run OEE's parser on the database. These steps invoke the OmniFind Analytics Edition Natural Language Processing required.

4. Run indexing on the database.

Now you have the basics for rule creation using the Rule Editor application. The difficulty is in deciding which rules to configure for each dataset, and how to configure those. The only way to get better at that is to dive in and start thinking. In the case study chapters in this book, we provide numerous examples of rules applied to real datasets. This should give you an excellent start.

Alerting System

In this chapter, we provide an overview of the OmniFind Analytics Edition Alerting System and its main functionalities. This chapter also provides detailed guidelines on how to set up alerts. We use a sample database from National Highway Traffic Safety Administration (NHTSA) when describing the detailed steps.

We cover the following topics:

- ▶ Alerting System functional overview
- ▶ Setting up alerts
 - Setting up Increase Detection
 - Setting up Correlation Detection
- ▶ Batch processing and result analysis

9.1 Alerting System functional overview

The Alerting System is a Web-based application that focuses on finding problems in a dataset by checking for an increase in the occurrence of particular keywords and combinations of highly-correlated keywords and categories. It performs two kinds of alert analysis:

- ▶ Increase Detection
- ▶ Correlation Detection

Increase Detection is the analysis process that detects the increasingly used keywords (including phrases), from a large number of keywords. The Increase Detection function allows users to monitor and analyze increase in occurrence of different keywords belonging to different categories of a dataset. The Text Miner also has an increase analysis function which can analyze up to 50 keywords in graphic format that shows increases and decreases. The Alerting System can analyze up to 20,000 keywords.

Correlation Detection is the analysis process that detects the combinations of keywords and subcategories that are highly correlated to each other. Although the Text Miner correlation analysis function can analyze up to 100 keywords along the vertical axis x 100 keywords and along the horizontal axis in a two-dimensional map, the Alerting System can analyze up to 1,000 keywords x 12,000 keywords.

9.2 Setting up alerts

Setting up the Alerting System can be summarized as follows:

1. Select the database to be used for alert extraction.
2. Choose the analysis type, either Increase Detection or Correlation Detection.
3. If you select Increase Detection, then you have to choose a category for which you want detect the increase and set up the parameters for detection of increase such as maximum number of alerts to be detected and maximum number of keywords to be analyzed. The Increase Detection Alert Results would show a list of increased keywords that were extracted in accordance with the categories and parameters chosen.

For example, for the NHTSA database, if you chose the category to be analyzed as Components, and select the maximum alerts to be 100, then the Increase Detection function would detect top 100 keywords falling under the Components category for which there has been a notable increase in frequency. In essence, it would list the 100 components for which there has been an increase in complaints over a particular period. It is possible to

connect to the Time Series view in Text Miner from each of these extracted keywords to analyze the exact increase in frequency.

4. If you select Correlation Detection, you have to choose multiple categories to be compared and set up the parameters for comparison such as maximum number of alerts to be detected and maximum number of keywords to be analyzed. The Correlation Detection function would compare the keywords or subcategories under the categories you chose to compare and analyze the correlation between each of these. The Correlation Detection Alert Results would show a list of combinations of highly-correlated keywords and categories that were extracted in accordance with the categories and parameters chosen.

For example, for the NHTSA database, if we chose to compare Component¹, and Noun Sequence² categories, and select the maximum alerts to be 100, then the Correlation Detection function would detect the top 100 combinations of components of complaints and the components which are highly correlated with each other. You can also connect to the Text Miner Docs view of relevant documents for each of these combinations to analyze the individual documents.

We discuss setting up Increase Detection and Correlation Detection in detail in this section.

9.2.1 Setting up Increase Detection

The Increase Detection function is based on the principle of ranking of keywords with the increased frequency of appearance on the latest date. The Text Miner performs the Increase Detection using the Delta view, but the number of keywords or subcategories that can be analyzed is very limited compared to the exhaustive Increase Detection analysis capabilities of the Alerting System.

In this section, we set up Increase Detection for a sample NHTSA complaint database. We set up an alert to analyze the most frequently occurring problems with different components in motor vehicles. So, in this case, we select Components as the category to be analyzed. The Components category categorizes documents based on the components or parts of vehicles found in the customer complaints. The increase in occurrence of a particular component keyword in the customer complaints would indicate a specific recurring problem with particular component. And the Increase Detection alert would enable you to identify the top 200 components for which there has been an increase in frequency of occurrence.

¹ The Component category categorizes documents based on the different component descriptions entered by the users in the complaints.

² The Noun Sequence category categorizes documents based on the noun sequences extracted from the customer complaints in the form of free text.

The steps involved in setting up an Increase Detection alert are as follows:

1. Launch the Alerting System from a browser:
`http://<Server Name:Port Number>/TAKMI_ALERT`
For our example, we enter:
`http://localhost:9080/TAKMI_ALERT`
2. Select the database for which you want to set up an alert. See Figure 9-1.
For our scenario, we select **NHTSA_latest**, which is the dataset containing the NHTSA customer complaints.

ALERTING SYSTEM - OmniFind Analytics Edition

Select Database ?

Please select a database from the following database list.

Database name
INDEXED_DATA_SAMPLE_EN
NHTSA_EN
NHTSA_latest

Figure 9-1 Database select window

3. Select **Increase Detection** as the analysis type. See Figure 9-2.

ALERTING SYSTEM Database: NHTSA latest

Analysis Type Selection ?

TOP PAGE

Please select the analysis type.

Analysis Type	Status
Increase Detection	Editable
Correlation Detection	Editable

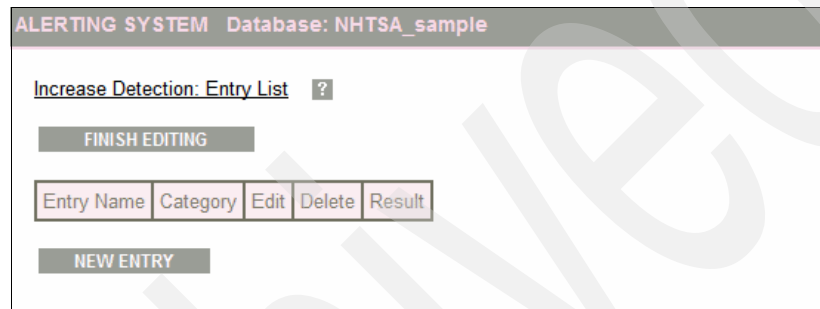
Figure 9-2 Analysis Type selection window

Due to data integrity reasons, the system allows only one person to configure Increase Detection for a database at any time. If the status field does not show Editable, clicking the link for the analysis type will present a dialog that asks whether you want to interrupt the edit. Click **OK** to cancel the settings currently being edited by another user and start editing.

4. Create a new increase alert entry.

For our example, we want to set up an early detection alert to detect issues or defects with the components of vehicles. Follow these steps to set up a new alert:

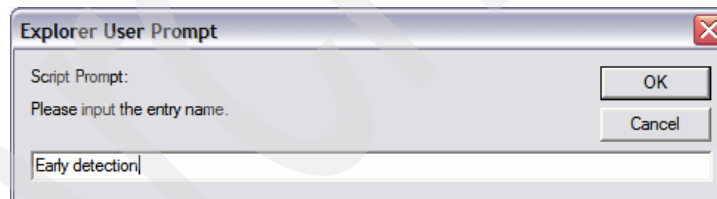
- a. In the Increase Detection window, click **NEW ENTRY**. See Figure 9-3.



The screenshot shows a window titled "ALERTING SYSTEM Database: NHTSA_sample". Inside, there is a section "Increase Detection: Entry List" with a question mark icon. Below this is a "FINISH EDITING" button. Underneath is a table with five columns: "Entry Name", "Category", "Edit", "Delete", and "Result". Below the table is a "NEW ENTRY" button.

Figure 9-3 Increase Detection entry window

- b. Enter the new alert entry name. For our example, we enter Early detection. See Figure 9-4.



The screenshot shows a dialog box titled "Explorer User Prompt". It contains a "Script Prompt:" label and the text "Please input the entry name.". Below this is a text input field containing the text "Early detection". To the right of the input field are two buttons: "OK" and "Cancel".

Figure 9-4 Enter alert entry name

- c. Click **OK**.

The Increase Detection window should now look like Figure 9-5.

ALERTING SYSTEM Database: NHTSA latest

Increase Detection: Entry List ?

FINISH EDITING

Entry Name	Category	Edit	Delete	Result
Early detection	Not set	CATEGORY PARAMETERS	DELETE	NOT ANALYZED

NEW ENTRY

Figure 9-5 Increase Detection entry list window

Note: In Increase Detection, you specify a keyword group and subcategory group as the target of detection. This category and a set of associated parameters are collectively managed as an *Entry*.

5. Select category settings for the newly created alert entry.

We need to select a category from the category tree to specify the keywords and subcategories that are to be subjected to the Increase Detection analysis.

To select the category:

- a. Click the **CATEGORY** link, and the window in Figure 9-6 displays.
- b. Click **SET TO CATEGORY** next to the selected category.

ALERTING SYSTEM Database: NHTSA latest

Increase Detection: Edit Category ?

BACK

Entry Name ?	Early detection
Category ?	Components

Category Tree

Part of Speech

SET TO CATEGORY

Phrase Constituent

SET TO CATEGORY

CAMPAIGN NUMBER

SET TO CATEGORY

CITY

SET TO CATEGORY

COMPONENT DESCRIPTION

SET TO CATEGORY

CRASH

SET TO CATEGORY

Components

SET TO CATEGORY

Customer's Voice

SET TO CATEGORY

DEATHS

SET TO CATEGORY

Figure 9-6 Increase Detection category selection window

c. Click **BACK**.

For our example, we select the Components category, so that the Increase Detection function would detect the increase in frequency of the component keywords or subcategories listed under the Components category.

Figure 9-7 shows the components category being set for the Early detection entry.

ALERTING SYSTEM Database: NHTSA_sample

Increase Detection: Entry List ?

FINISH EDITING

Entry Name	Category	Edit	Delete	Result
Early detection	components	CATEGORY PARAMETERS	DELETE	NOT ANALYZED

NEW ENTRY

Figure 9-7 Component category set for the Early detection alert entry

6. Set the parameters for the newly created Increase Detection alert entry:
 - a. Click **PARAMETERS** next to the alert entry.

Figure 9-8 shows the parameters you can set up for the Early detection alert entry.

ALERTING SYSTEM Database: NHTSA latest

Increase Detection: Edit Parameters ?

SAVE **CANCEL**

Entry Name ?	Early detection
Category ?	Components
Maximum number of alerts ?	<input type="radio"/> 10 <input type="radio"/> 20 <input type="radio"/> 50 <input checked="" type="radio"/> 100 <input type="radio"/> 200
Maximum number of keywords ?	<input checked="" type="radio"/> 100 <input type="radio"/> 1000 <input type="radio"/> 5000 <input type="radio"/> 10000 <input type="radio"/> 20000
Minimum frequency ?	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5
Decaying factor ?	<input type="radio"/> 0.70 <input type="radio"/> 0.75 <input type="radio"/> 0.80 <input checked="" type="radio"/> 0.85 <input type="radio"/> 0.90
Time scale ?	<input type="radio"/> Month <input checked="" type="radio"/> Week <input type="radio"/> Day

Done Local intranet

Figure 9-8 Increase Detection parameter selection window

- b. Set the following parameters:
 - Entry name: Specify the alert entry name. You can rename it at this point.
 - Maximum number of alerts: Specify the maximum number of alerts to be detected. Detection is not based on a threshold, but instead, the keywords or subcategories whose increase indicator is ranked within the maximum number are detected.

For our example, we set it to 100. This means that the top 100 keywords or subcategories in Components category which the rate of increase in frequency of use has been notably rising will be alerted.

- **Maximum number of keywords:** Specify the maximum number of keywords or subcategories to be analyzed.

For our example, we set to 100. This means that the total number of keywords to be analyzed is 100. The keywords and subcategories with a high rate of increase in frequency of use are detected among 100 keywords and subcategories of the Components category.

- **Minimum frequency:** Although a collection of documents that is retrieved under the conditions described above is attached to each alert, alerts with only a few documents might be meaningless; therefore, specify the minimum number of documents required to detect alerts. Alerts® will not be returned if the number of applicable documents is below the set value.

For our example, we set it to 3. This is for testing only. This is set so that only alerts with more than 3 documents attached to it can be considered.

- **Decaying factor:** In Increase Detection, keywords and subcategories that “are relatively more frequently used recently than the past average frequency” are detected, and this parameter specifies how old the data should be when obtaining the past average frequency. The past value weighs more as the value becomes larger. For example, when the decaying factor is 0.8 and the time scale is set to month, the frequency obtained two months ago weighs 0.8 times more than the frequency obtained in the previous month when calculating the average.

For our example, we set it to 0.85

- **Time scale:** From monthly, weekly, and daily, select one for acquisition of frequency data when analyzing the time-series frequency of use of keywords and subcategories. Select a long time scale if you wish to analyze a slow increase in frequency or to analyze keywords that are not used frequently. Select a short time scale if you wish to analyze a short-term increase in frequency or to analyze keywords that are used moderately or quite frequently.

For our example, we set the Time scale as Week.

c. Click **SAVE** to save the parameter setting.

Figure 9-9 shows the Early detection alert entry. Notice that the Result shows NOT ANALYZED yet. In later steps, we show you how to start the alert analysis and view the results.

If you made a mistake at this stage, you can go back and re-select the category, or re-set the parameter, or delete the alert entry.

ALERTING SYSTEM Database: NHTSA_sample

Increase Detection: Entry List ?

FINISH EDITING

Entry Name	Category	Edit	Delete	Result
Early detection	components	CATEGORY PARAMETERS	DELETE	NOT ANALYZED

NEW ENTRY

Figure 9-9 Early detection alert entry editing

7. Click **FINISHED EDITING** to save the alert entry and go back to the main alert page, where you can specify to set Increase Detection or Correlation Detection.

9.2.2 Setting up Correlation Detection

Correlation Detection is the process of extraction of combinations of keywords and subcategories that are highly correlated to each other. The 2D Map view in Text Miner performs similar Correlation Detection analysis, but the volume of data that can be analyzed using Correlation Detection with Alerting System is much larger than the Text Miner.

In this section we set up Correlation Detection for the NHTSA complaint database. We set up a correlation alert to analyze the combinations of keywords from the Component Description category and the Noun Sequence category to find out the combinations of motor vehicle parts that are highly correlated with the customer complaints.

Some of these alerts might be pointing to something very obvious, such as airbags getting deployed during accidents. But we are also able to discover alerts saying that there is a high correlation between airbag deployment and eye injuries, and when we drilled down to the actual documents, there are indeed cases of defective airbags causing eye and face injuries to the passengers while deployment.

The steps involved in setting up a Correlation Detection alert are as follows:

1. Launch the Alerting System from a browser:
`http://<Server Name:Port Number>/TAKMI_ALERT`
For our example, we enter:
`http://localhost:9080/TAKMI_ALERT`
2. Select the database for which you want to set up an alert.
For our example, we select **NHTSA_latest**, which is the dataset containing the NHTSA customer complaints.
3. Select **Correlation Detection** as the analysis type for the alert.
Due to data integrity reason, the system allows only one person to configure Correlation Detection for a database at any time. If the status field does not show Editable, clicking the link for the analysis type will present a dialog that asks whether you want to interrupt the edit. Click **OK** to cancel the settings currently being edited by another user and start editing.
4. Create a new Correlation Detection alert entry.
 - a. Click **NEW ENTRY**. See Figure 9-4.
 - b. Enter the new alert entry name and click **OK**.
For our example, we enter Parts Finder.

Figure 9-10 shows what we have now for the Parts Finder alert entry. As mentioned earlier, this alert will be used to detect the different combinations of vehicle components involved in customer complaints that have high correlations with other parts or problems.

ALERTING SYSTEM Database: NHTSA latest

[Correlation Detection: Entry List](#) ?

FINISH EDITING

Entry Name	Compared category	Content category	Edit	Delete	Result
Parts Finder	Not set	Not set	CATEGORY PARAMETERS	DELETE	NOT ANALYZED

NEW ENTRY

Figure 9-10 Correlation Detection entry list window

5. Select category settings for the newly created alert entry:
 - a. Click **CATEGORY**.
 - b. From the category tree (Figure 9-11), click **COMPARED CATEGORY** next to the category you want to compare. Click **ADD TO CONTENTCATEGORY** next to the category you want to add to the content category.

Compared and content categories are categories for which the keywords and subcategories belonging to each would be compared with the other to find correlations. The Correlation Detection function compares a predefined number of keywords or subcategories belonging to the compared category with a predefined number of keywords or subcategories belonging to the content categories, and records the correlations between each of these. The top specified number of combinations with highest correlations are then listed as correlation alerts. Also, up to two categories can be specified as the content categories.

Correlation Detection: Edit Category
?

BACK

Entry Name ?	Parts Finder	Category Tree	
Compared category ?	COMPONENT DESCRIPTION	<div> <div> Part of Speech </div> <div>COMPARED CATEGORY</div> <div>ADD TO CONTENT CATEGORY</div> </div>	
Content category (1 or 2 categories) ?	Noun Sequence REMOVE	<div> <div> Phrase Constituent </div> <div>COMPARED CATEGORY</div> <div>ADD TO CONTENT CATEGORY</div> </div>	
		<div> <div> CAMPAIGN NUMBER </div> <div>COMPARED CATEGORY</div> <div>ADD TO CONTENT CATEGORY</div> </div>	
		<div> <div> CITY </div> <div>COMPARED CATEGORY</div> <div>ADD TO CONTENT CATEGORY</div> </div>	
		<div> <div> COMPONENT </div> <div>COMPARED CATEGORY</div> <div>ADD TO CONTENT CATEGORY</div> </div>	

Figure 9-11 Correlation Detection category selection window

- For our example, we choose Component Description as the compared category and Noun Sequence as the content category. So, the different keywords or subcategories under the Component Description category would be compared with keywords or subcategories under the Noun Sequence category. Notice that to get to the Noun Sequence, you must expand **Phrase Constituent** in the category tree.
- c. Click **BACK**.
 6. Set the parameters for the newly created alert entry:
 - a. Click **PARAMETERS**.
 - b. Set the parameters shown in Figure 9-12.

ALERTING SYSTEM Database: NHTSA latest

Correlation Detection: Edit Parameters ?

SAVE CANCEL

Entry Name ?	Parts Finder
Compared category ?	COMPONENT DESCRIPTION
Content category ?	Noun Sequence
Maximum number of alerts ?	<input type="radio"/> 10 <input type="radio"/> 20 <input type="radio"/> 50 <input checked="" type="radio"/> 100 <input type="radio"/> 200
Maximum number of keywords in compared category ?	<input type="radio"/> 50 <input type="radio"/> 100 <input type="radio"/> 200 <input checked="" type="radio"/> 500 <input type="radio"/> 1000
Maximum number of keywords in content category ?	<input type="radio"/> 50 <input type="radio"/> 100 <input type="radio"/> 200 <input type="radio"/> 500 <input checked="" type="radio"/> 1000
Maximum number of keyword pairs in content category ?	<input type="radio"/> 100 <input type="radio"/> 500 <input type="radio"/> 1000 <input type="radio"/> 5000 <input checked="" type="radio"/> 10000
Minimum frequency ?	<input type="radio"/> 1 <input type="radio"/> 2 <input checked="" type="radio"/> 3 <input type="radio"/> 4 <input type="radio"/> 5
Confidence coefficient ?	<input checked="" type="radio"/> 0.80 <input type="radio"/> 0.85 <input type="radio"/> 0.90 <input type="radio"/> 0.95 <input type="radio"/> 0.98

Figure 9-12 Correlation Detection parameter setup window

Parameters to set up include:

- **Entry name:** Specify the name of the alert entry. You can rename it at this point.
- **Maximum number of alerts:** Specify the maximum number of alerts to be detected. Alerts are detected as combinations of keyword or subcategory of the compared category, and keyword or subcategory (or two subcategories) of the content category, and returned in the order of strength of correlation between the compared category and the content category up to the number specified in this setting.

For our example, we set 100. This means that the top 100 combinations of highly-correlated keywords and subcategories from Component Description and Noun Sequence will be returned.

- **Maximum number of keywords in compared category:** Set the maximum number of compared category keywords or subcategories to be used in the analysis.

For our example, we set it to 500. This means when the 500 most frequently used components in the Component Description category will be the subject of the analysis.

- **Maximum number of keywords in content category:** Set the maximum number of content category keywords or subcategories to be used in the analysis.

For our example, we set it to 1000. This means that the 1000 most frequently mentioned noun sequences in the Noun Sequence category will be the subject of the analysis.

- **Maximum number of keyword pairs in content category:** When two content categories are specified, then set the maximum number of pairs consisting of the keywords and subcategories of the first content category and the keywords and subcategories of the second content category.

This is only applicable if we selected multiple content categories. For our example, we leave the default value as is.

If, for example, we had selected two content categories, Noun Sequence and the City category, and set this number to 500, then the Correlation Detection function would compare at most 500 pairs of combinations of keywords or subcategories of both Noun Sequence and City category with the keywords or subcategories of the compared category, Component Description.

- **Minimum frequency:** Although a collection of documents that is retrieved under the conditions described above is attached to each alert, having alerts with only a few documents might be meaningless; therefore, specify the minimum number of documents required to detect alerts. Alerts will not be returned if the number of applicable documents is below the set value.

For our example, we set it to 3. This is a small number for testing only.

- **Confidence coefficient:** This is a parameter used when statistically calculating a correlation value (correlation strength). If the coefficient is set high, a relatively large number of alerts having the sufficient number of applicable documents for calculating correlation strength will be returned, and if it is set low, a large number of alerts with the slightest possibilities will be returned even if there is not a sufficient number of applicable documents for calculating correlation strength.

For our example, we this parameter to 0.8.

c. Click **SAVE** to save the parameter setting.

7. Click **FINISH EDITING** to complete the Correlation Detection alert entry configuration.

9.3 Batch processing and result analysis

This section describes in detail the steps involved in running batch processing for Increase Detection and Correlation Detection and analyzing the results.

9.3.1 Batch processing

After the alert settings are successfully completed via the Alerting System application, the next step would be to run the batch process to generate the results of the newly created (or edited) alert. Every time an alert is created or edited, it is required to run the batch process to generate or update the alert analysis results. There are two separate processes that should be run for Increase Detection and Correlation Detection alerts. Running the batch process would generate an analysis result report file in the <DATABASE_DIR>\alerting\batch directory.

Note: <TAKMI_HOME> is the directory where OmniFind Analytics Edition is installed. For a default installation, <TAKMI_HOME> is set to C:\Program Files\IBM\takmi

<DATABASE_DIR> refers to your database directory. For our sample, we are using <TAKMI_HOME>\database\NHTSA_SAMPLE as the DATABASE_DIR.

For generating the analysis results for an Increase Detection alert, open a command window (on Windows) or shell (on AIX) and execute the command as shown in Example 9-1.

Example 9-1 Usage of increase alert batch process command

Windows:

```
> takmi_alert_increase.bat <DATABASE_NAME>  
<MAXIMUM_ANALYSIS_TIME_BY_MINUTE> <JAVA_HEAP_SIZE_BY_MB>
```

AIX:

```
> takmi_alert_increase.sh <DATABASE_NAME>  
<MAXIMUM_ANALYSIS_TIME_BY_MINUTE> <JAVA_HEAP_SIZE_BY_MB>
```

The arguments have the following meanings:

- ▶ <DATABASE_NAME>: The database name defined in the global_config/database_entries/database_entry name of global_config.xml.
- ▶ <MAXIMUM_ANALYSIS_TIME_BY_MINUTE>: Specify the maximum analysis time in units of minutes.

- ▶ `<JAVA_HEAP_SIZE_BY_MB>`: Specify the Java heap size for analysis in units of MB.

Note: The language processing and index creation must be completed within the database where Increase Detection or Correlation Detection is to be run. In addition, the Alerting System Web application must not be in operation.

To generate the analysis results for a Correlation Detection alert, open a command window (on Windows) or shell (on AIX) and execute the command as shown in Example 9-2.

Example 9-2 Usage of correlation alert batch process command

Windows:

```
> takmi_alert_correlation.bat <DATABASE_NAME>  
<MAXIMUM_ANALYSIS_TIME_BY_MINUTE> <JAVA_HEAP_SIZE_BY_MB>
```

AIX:

```
> takmi_alert_correlation.sh <DATABASE_NAME>  
<MAXIMUM_ANALYSIS_TIME_BY_MINUTE> <JAVA_HEAP_SIZE_BY_MB>
```

The meaning of the arguments are as follows:

- ▶ `<DATABASE_NAME>`: The database name defined in the `global_config/database_entries/database_entry` name of `global_config.xml`.
- ▶ `<MAXIMUM_ANALYSIS_TIME_BY_MINUTE>`: Specify the maximum analysis time in units of minutes.
- ▶ `<JAVA_HEAP_SIZE_BY_MB>`: Specify the Java heap size for analysis in units of MB.

9.3.2 Increase Detection results

On completion of the Increase Detection batch processing, the analysis results for the alerts created (or edited) would be displayed in the Alerting System application. To view the results for the alerts, follow these steps:

1. Launch the Alerting System from a browser.

For our use case, we enter:

`http://localhost:9080/TAKMI_ALERT`

2. Select the database for which you had set up an Increase Detection alert.

For our scenario, we use `NHTSA_latest`, which is the dataset containing the NHTSA customer complaints. We had created an alert “Early detection” for this database in section 9.2.1, “Setting up Increase Detection” on page 223.

3. Select **Increase Detection**.
 4. Click **VIEW** to see the analysis results (see Figure 9-13).
- The VIEW link is generated during the batch processing.

ALERTING SYSTEM Database: NHTSA latest

Increase Detection: Entry List ?

FINISH EDITING

Entry Name	Category	Edit	Delete	Result
Early detection	Components	CATEGORY PARAMETERS	DELETE	VIEW

NEW ENTRY

Figure 9-13 Increase Detection entry list window after batch processing

The Increase Detection alert result window that lists the alert results would contain the complete alert configuration details, the last batch processing details, and the alert results. See Figure 9-14.

ALERTING SYSTEM Database: NHTSA latest

Increase Detection: Alert Results ?

Entry Name	Early detection	Date	Time setting (by minute)	Time (by minute)	Exit status	Threshold (yellow)	Threshold (orange)	Threshold (red)
Early detection	Components	13 Oct 2007	5	2	FINISHED	5.0	10.0	20.0

NEW ENTRY

Figure 9-14 Increase Detection alert results

The Increase Detection alert results would list the top 100 keywords or subcategories in the order of maximum increase in frequency of occurrence, since in this case we have selected 100 as the maximum number of alerts value. Each entry in the results table would have a Jump link, which would open up the Text Miner Time Series view for the particular keyword or subcategory for which the alert is generated.

So in our case, the top 2 results are “replace....bolt” and “missing....bolt”. Clicking the **Jump** link for these would take us to the Text Miner Time Series view for these keywords (see Figure 9-15). Examining the time series charts, it shows that there has been an increase of occurrence of the particular keyword. Analyzing these results further using the Text Miner, we find that the complaints about recurring problems are related to the front seat bolts in a particular model that causes the seat to fly back into extended cab.

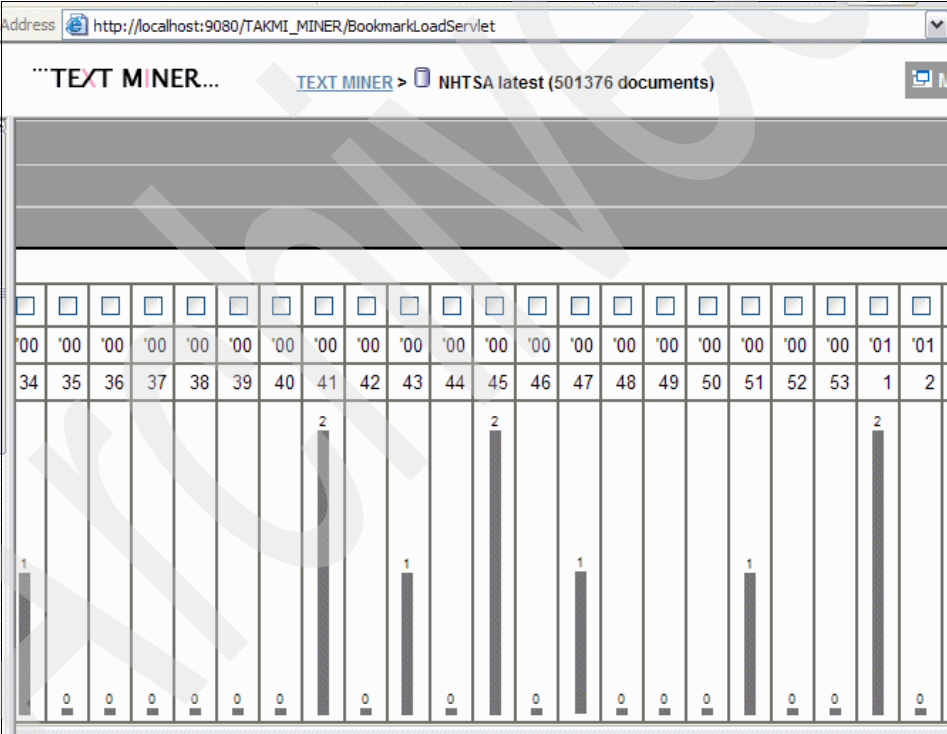


Figure 9-15 Time series view for the Increase Detection alert result

9.3.3 Correlation Detection results

On completion of the Correlation Detection batch processing, the analysis results for the alerts created (or edited) would be displayed in the Alerting System application. To view the results for the alerts:

1. Launch the Alerting System from a browser.
For our use case, we enter:
`http://localhost:9080/TAKMI_ALERT`
2. Select the database for which you had set up an Correlation Detection alert.
For our scenario, we use NHTSA_latest, which is the dataset containing the NHTSA customer complaints. We had created an alert “Parts Finder” for this database in section 9.2.2, “Setting up Correlation Detection” on page 230.
3. Select **Correlation Detection**.
4. Click VIEW to see the analysis results (see Figure 9-16).
The VIEW link is generated during the batch processing.

ALERTING SYSTEM Database: NHTSA latest

Correlation Detection: Entry List ?

FINISH EDITING

Entry Name	Compared category	Content category	Edit	Delete	Result
Parts Finder	COMPONENT DESCRIPTION	Noun Sequence	CATEGORY PARAMETERS	DELETE	VIEW

NEW ENTRY

Figure 9-16 Correlation Detection entry list window after batch processing

Figure 9-17 shows the Correlation Detection alert result window.

ALERTING SYSTEM Database: NHTSA latest			
Correlation Detection: Alert Results			
Entry Name	Parts Finder	Date	13 Oct 2007
Compared category	COMPONENT DESCRIPTION	Time setting (by minute)	5
Content category	Noun Sequence	Time (by minute)	Less than 1 minute
Maximum number of alerts	100	Exit status	FINISHED
Maximum number of keywords in compared category	500	Threshold (yellow)	5.0
Maximum number of keywords in content category	1000	Threshold (orange)	10.0
Maximum number of keyword pairs in content category	10000	Threshold (red)	20.0
Minimum frequency	3		
Confidence coefficient	0.8		

Figure 9-17 Correlation Detection alert results

The result window contains the complete alert because we have chosen the maximum number of alerts as 100. The results would list the top 100 keyword (subcategory) combinations from the compared and content categories in the order of correlation. Each entry in the results table would have a Jump link, which would open up the Text Miner Docs view for the particular combination of keywords or subcategories for which the alert is generated.

The top result in the alert results table says that there is a strong correlation between the keywords “SERVICE BRAKES, AIR:SUPPLY:COMPRESSOR” and “air compressor.” Clicking the **Jump** link takes us to the Text Miner Docs view for the particular combination of keywords. The documents listed under this combination are all complaints about recurring problems related to a particular model of air compressors used in trucks. The correlation alert helps to detect the problems before they reach a breaking point.

Case studies

This chapter describes case studies based on the analysis performed for different types of data. The purpose of these case studies is to demonstrate the capabilities and features of IBM OmniFind Analytics Edition (OAE) and to show you how you can use it to analyze your data.

We cover the following case studies:

- ▶ Case study: Technical Help Desk
- ▶ Case study: NHTSA
- ▶ Case study: e-mail database

At the end of the chapter, we also cover how to apply new rules and customized dictionary if you made changes to your dictionary and rules.

Note: While this chapter guides you through building a general-purpose text analytics and text mining solution, we assume that you have read and understand the key concepts that are discussed in previous chapters. If you have not done so, we highly recommend that you read them and become familiar with the contents of those chapters prior to reading these case studies.

10.1 Case studies overview

Each case study represents data from different organizations, and each of them showcases different ways to analyzing customer data using OmniFind Analytics Edition. We show the work that goes into text analysis and text mining the data. The same work would be virtually impossible without the help of the robust text mining tool, OmniFind Analytics Edition.

Case study overview for Technical Help Desk

For the first case study, we obtain data from a Technical Help Desk which includes complaints and issues from callers. The company goal is to use this information to detect common complaints and identify unsatisfactory product attributes so that the company can initiate remedies for these problems before it is too late.

This case study demonstrates how to perform a generic analysis of data *without knowing what to look for and where to look*.

For this case study, we cover:

- ▶ The steps required to set up the database (an overview)
- ▶ Basic analysis (without knowing what and where to look for data)
- ▶ Root cause analysis of problem

The functions and features of OmniFind Analytics Edition that we showcase in this case study include:

- ▶ Category tree
- ▶ Part of Speech (looking for high frequency words)
- ▶ Category view
- ▶ Phrase Constituent (looking for high frequency phrases)
- ▶ Search condition buildup
- ▶ Voice of customers (looking for high correlation in Category view)
- ▶ Rules customization (creating rules to ignore specific data entries so it is easier to perform analysis on the remaining relevant data)

Case study overview for NHTSA

For the second case study, we obtain data from the National Highway Traffic Safety Administration (NHTSA). The goal is to use this information to analyze complaints and discover common concerns related to specific vehicle makers, models, and components.

This use case demonstrates how to perform analysis of data in areas of special concern (for our example, any complaints that are fire related).

The functions and features of OmniFind Analytics Edition that we showcase in this case study include:

- ▶ Dictionary customization (creating a new category to group all related keywords under that category for analysis)
- ▶ Search condition buildup
- ▶ 2D Map view (associating problem areas with components under investigation and showing high correlation problem area)
- ▶ Part of speech with Category view (looking for high frequency and high correlation keywords)

Case study overview for e-mail databases

For the third case study, we obtain data from our internal mail databases. The goal is to use this information to analyze any concerns or issues related to a specific area (in our example, it is a Visa requirement).

This case study demonstrates how to perform analysis of data when concerning a specific subject matter (for example, a project, an action, or a procedure).

The functions and features of OmniFind Analytics Edition that we showcase in this case study include:

- ▶ Customizing dictionary (to categorize area of interests for analysis)
- ▶ Customizing rules (to fine tune analysis)

For each case study, we provide an overview of the business scenario, where the data comes from and what they represent. Then we provide details as how to do text analytics with these data. Each case study is unique and we recommend reading through all of them to get a good picture of how you can set up and analyze your data.

10.2 Case study: Technical Help Desk

This case study is based on the analysis performed for the internal Technical Help Desk of an information technology company. The purpose of this case study is to demonstrate the capabilities and features of IBM OmniFind Analytics Edition and to illustrate ways to use the product for data analysis.

We demonstrate the use of Category tree, Category view, Part of speech, phrase constituents, search buildups. In addition, we present a need for rule customization and how it can be done.

10.2.1 Overview of the business scenario

The case study involves a technology company, which sells hardware, software, and services to its clients. The company has an internal Technical Help Desk, which is designed to address all the technical problems related to connectivity, workstations, hardware, internal applications, passwords, and security issues faced by the employees. The company needs a text analysis solution that streamlines and advances the collection, review, and analysis of information housed in the Technical Help Desk applications to detect and proactively address escalating end user issues, to build more effective Web support, and in turn reduce the number of calls to lower the overall operation cost.

The Technical Help Desk department uses a database managed application that records all the details pertaining to different complaints logged by the employees, the status of these complaints, and resolutions provided for each of these. They wanted to use this information to detect the common complaints and identify unsatisfactory product attributes so that they could initiate remedies for these problems before it is too late. We demonstrate how OmniFind Analytics Edition is able to cater to their requirements and use their existing Technical Help Desk data to discover useful information that can help them increase the Technical Help Desk efficiency and reduce their operational costs.

Company goal: Detect common complaints and identify unsatisfactory product attributes so that the company can initiate remedies for these problems before it is too late.

10.2.2 Technical Help Desk database details

For this case study, we use the Technical Help Desk database, which contains all the complaints and issues logged by the users or employees. This database basically contains the different technical complaints related to hardware problems such as complaints about PCs, software problems such as internal portals not working, Lotus mail database not responding, network outages, and any other technical problems faced by the employees. The Technical Help Desk analysts use a database managed application for logging the calls and recording the progress and resolution details of the logged calls, and this data is stored in a relational database.

In Table 10-1, we identify the schema of the Technical Help Desk database.

Table 10-1 Schema for the Technical Help Desk database

No	Column name	Data type and description
1	PROBLEM_ID	CHAR(9) Unique sequence number for every problem reported
2	PROBLEM_CODE	CHAR(9) Predefined codes for different categories of problems
3	PROBLEM_TYPE	CHAR(25) Type of problem
4	SEVERITY	CHAR(1) Severity of the problem entered by the user(1-3)
5	ORIGINAL_SEVERITY	CHAR(1) Original severity of the problem(1-3)
6	PROBLEM_ABSTRACT	CHAR(256) Abstract of the problem
7	OCCURRED_DATE	CHAR(8) Date on which the problem occurred
8	OCCURRED_TIME	CHAR(8) Time at which the problem occurred
9	OPEN_DATE	CHAR(8) Date on which the problem ticket is opened
10	OPEN_TIME	CHAR(8) Time at which the problem ticket is opened
11	CAUSE_CODE	CHAR(25) Pedefined generic categories of problem causes
12	CLOSE_DATE	CHAR(8) Date on which the problem ticket is closed
13	CLOSE_TIME	CHAR(8) Time at which the problem ticket is closed
14	DURATION	CHAR(10) Duration taken for resolving the problem

No	Column name	Data type and description
15	LOCATION_NAME	CHAR(25) Location at which the problem occurred
16	GROUP_ID	CHAR(40) Group from which problem was reported
17	RESOLVER_ID	CHAR(40) Analyst in charge of providing resolution
18	RESOLVER_GROUP	CHAR(40) Group to which the resolver belongs
19	FIRST_CALL_ID	CHAR(10) Telephone number of the affected employee
20	FIRST_CONTACT_ID	CHAR(25) Name of the affected employee
21	FIRST_LOCATION_ID	CHAR(15) Location of the affected employee
22	SOLUTION_ID	CHAR(9) Unique Id of the solution provided
23	ITEM	CHAR(25) Category to which the problem belongs
24	COMPONENT	CHAR(25) Another categorization of the problem
25	MODULE	CHAR(25) Predefined module to which problem belongs
26	SYSTEM	CHAR(6) System which encountered the problem
27	TEAM	CHAR(6) Team maintaining the above system
28	USER_ID	CHAR(25) Id of the maintaining group
29	REPORTER_GROUP	CHAR(25) Group of the reporter
30	REPORTER_ID	CHAR(15) Name of the reporter
31	CURR_TARGET_DATE	CHAR(8) Target date for resolution

No	Column name	Data type and description
32	CURR_TARGET_TIME	CHAR(8) Target time for resolution
33	ACCOUNT_ID	CHAR(15) Affected account
34	CUSTOMER_ID	CHAR(6) Different subdivisions facing the problem
35	SOLVED_DATE	CHAR(8) Actual date of resolution
36	SOLVED_TIME	CHAR(8) Actual time of resolution
37	DESCRIPTION	CHAR(2048) Description of the complaint. Free text data that represents a complaint. This free text description will be broken down into nouns, adjectives, verbs or other parts of speech by the language processing annotators in OmniFind Analytics Edition. For the rest of the case study, we identify the contents of this column as unstructured source of data.
38	PROBLEM_RESULT	CHAR(256) Description of the result of the problem in the form of Free text data

The bold fields from Table 10-1, PROBLEM_ABSTRACT, DESCRIPTION, and PROBLEM_RESULT are free-formatted, unstructured data fields. The rest of the fields are structured fields.

10.2.3 Setting up the case study

Figure 10-1 provides a high level overview of the steps involved in setting up the database.

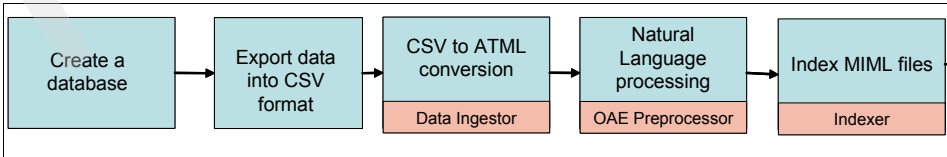


Figure 10-1 High level case study setup overview

To set up the database, you have to perform the following tasks:

1. Create a database.
2. Export data into CSV format.
3. Edit category tree.
4. Convert CSV file to ATML file.
5. Run natural language processing.
6. Index files.

Database creation

The first step would be to create a database. For this case study, we will create a database “IGA_HELP” and update the global_config.xml with the new database entry (Example 10-1).

Example 10-1 Entries in global_config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<global_config>
<params>
<param name="language" value="en"/>
</params>
<database_entries>
<database_entry name="INDEXED_DATA_SAMPLE_EN" path_type="relative"
path="databases/INDEXED_DATA_SAMPLE_EN"/>
<database_entry name="IGA_HELP" path_type="relative"
path="databases/IGA_SAMPLE"/>
</database_entries>
</global_config>
```

Refer to 4.2, “Database directory creation and registration” on page 58 for detailed steps on how to create a database.

Exporting data into CSV format

Once you have created the database, the next step is to convert the data to be analyzed into CSV (Comma Separated values) format. Most of the relational databases and spreadsheet applications support exporting data to CSV format. In our case, the Technical Help Desk application stores all the data in a IBM DB2 database. You can use DB2’s export program to export your target table data into CSV format. In this case, all the contents are stored in a table called IGA_STG, that resides in a database called IGA_HELP_DB, are exported into a CSV file called IGA.csv (Example 10-2).

Windows:

```
db2cmd (Launch a DB2 command window)
db2 connect to IGA_HELP_DB
db2 export to IGA.csv of del messages IGA.msg "select * from IGA_STG"
db2 terminate
```

Edit category tree

After the data is converted into CSV format, edit the `category_tree.xml` to relate each column in a CSV file to a particular category. Also, edit database settings in the `database_config.xml` file in order to define the category edited here as a standard item (see Example 10-3). Refer to 7.2.1, “Editing the category tree” on page 168 for more details on editing category tree and database configuration settings.

Example 10-3 Entries in `category_tree.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<category_tree>
.....
.....
<node id="202" path="PROBLEM_TYPE" name="PROBLEM_TYPE" features=""/>
<node id="203" path="SEVERITY" name="SEVERITY" features=""/>
<node id="204" path="ORIGINAL_SEVERITY" name="ORIGINAL_SEVERITY"
features=""/>
<node id="208" path="OPEN_DATE" name="OPEN_DATE" features=""/>
<node id="209" path="OPEN_TIME" name="OPEN_TIME" features=""/>
<node id="211" path="CAUSE_CODE" name="CAUSE_CODE" features=""/>
<node id="215" path="DURATION" name="DURATION" features=""/>
<node id="219" path="LOCATION_NAME" name="LOCATION_NAME" features=""/>
<node id="226" path="ITEM" name="ITEM" features=""/>
<node id="227" path="COMPONENT" name="COMPONENT" features=""/>
<node id="228" path="MODULE" name="MODULE" features=""/>
<node id="229" path="SYSTEM" name="SYSTEM" features=""/>
<node id="230" path="TEAM" name="TEAM" features=""/>
<node id="240" path="ACCOUNT_ID" name="ACCOUNT_ID" features=""/>
<node id="241" path="CUSTOMER_ID" name="CUSTOMER_ID" features=""/>
</category_tree>
```

After editing the category tree, edit the database settings to prevent the newly registered categories in the category tree from being edited by the Dictionary Editor tool. See Example 10-4.

```
<category_entries>
  <!-- Specifies subroot categories for system-reserved
categories. -->
  .....
  .....
  <category_entry name="reserved_by_system" value=".PROBLEM_ID"/>
  <category_entry name="reserved_by_system" value=".PROBLEM_CODE"/>
  <category_entry name="reserved_by_system" value=".PROBLEM_TYPE"/>
  <category_entry name="reserved_by_system" value=".SEVERITY"/>
  <category_entry name="reserved_by_system" value=".ORIGINAL_SEVERITY"/>
  <category_entry name="reserved_by_system" value=".OCCURRED_DATE"/>
  <category_entry name="reserved_by_system" value=".OCCURRED_TIME"/>
  <category_entry name="reserved_by_system" value=".OPEN_DATE"/>
  <category_entry name="reserved_by_system" value=".OPEN_TIME"/>
  <category_entry name="reserved_by_system" value=".CAUSE_CODE"/>
  <category_entry name="reserved_by_system" value=".CLOSE_DATE"/>
  <category_entry name="reserved_by_system" value=".CLOSE_TIME"/>
  <category_entry name="reserved_by_system" value=".DURATION"/>
  <category_entry name="reserved_by_system" value=".FIRST_CALL_ID"/>
  <category_entry name="reserved_by_system" value=".FIRST_CONTACT_ID"/>
  <category_entry name="reserved_by_system" value=".FIRST_LOCATION_ID"/>
  <category_entry name="reserved_by_system" value=".LOCATION_NAME"/>
  <category_entry name="reserved_by_system" value=".GROUP_ID"/>
  <category_entry name="reserved_by_system" value=".RESOLVER_ID"/>
  <category_entry name="reserved_by_system" value=".RESOLVER_GROUP"/>
  <category_entry name="reserved_by_system" value=".SOLUTION_ID"/>
  <category_entry name="reserved_by_system" value=".ITEM"/>
  <category_entry name="reserved_by_system" value=".COMPONENT"/>
  <category_entry name="reserved_by_system" value=".MODULE"/>
  <category_entry name="reserved_by_system" value=".SYSTEM"/>
  <category_entry name="reserved_by_system" value=".TEAM"/>
  <category_entry name="reserved_by_system" value=".USER_ID"/>
  <category_entry name="reserved_by_system" value=".REPORTER_GROUP"/>
  <category_entry name="reserved_by_system" value=".REPORTER_ID"/>
  <category_entry name="reserved_by_system" value=".CURR_TARGET_DATE"/>
  <category_entry name="reserved_by_system" value=".CURR_TARGET_TIME"/>
  <category_entry name="reserved_by_system" value=".ACCOUNT_ID"/>
  <category_entry name="reserved_by_system" value=".CUSTOMER_ID"/>
  <category_entry name="reserved_by_system" value=".SOLVED_DATE"/>
  <category_entry name="reserved_by_system" value=".SOLVED_TIME"/>
</category_entries>
```

CSV to ATML conversion

The data in CSV format must be converted into ATML, which is the standard input format for OmniFind Analytics Edition. See 4.5, “Generating ATML files” on page 70 for a comprehensive study on the CSV to ATML conversion.

Before running the Data Ingestor command, edit the file, `data_ingester_config_csv2atml.xml` to provide the CSV related column information.

Example 10-5 Entries in data_ingester_config_csv2atml.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ingester_config SYSTEM "data_ingester_config.dtd">
<!--
=====
    Configuration for Data Ingestor
=====
-->
<ingester_config>
.....
.....
<!--
=====
    Indexes of text columns. 1, 2, ... are available.
=====
-->
<param name="csv.column.text.indexes" multivalued="yes">
<value>37</value>
<value>38</value>
</param>
.....
.....
<!--
=====
    Index of the first row to be read. 1, 2, ... are available.
    If not specified, Ingestor will read all rows as data for TAKMI.
=====
-->
<param name="csv.row.firstindex" multivalued="no">
<value>1</value>
</param>
<!--
=====
    Path or names for each column in csv.
=====
```

```

-->
<param name="csv.column.names" multivalued="yes">
<value>.PROBLEM_ID</value>
<value>.PROBLEM_CODE</value>
<value>.PROBLEM_TYPE</value>
<value>.SEVERITY</value>
<value>.ORIGINAL_SEVERITY</value>
<value>.PROBLEM_ABSTRACT</value>
<value>.OCCUR_DATE</value>
<value>.OCCUR_TIME</value>
<value>.OPEN_DATE</value>
<value>.OPEN_TIME</value>
<value>.CAUSE_CODE</value>
<value>.CLOSE_DATE</value>
<value>.CLOSE_TIME</value>
<value>.DURATION</value>
<value>.FIRST_CALL_ID</value>
<value>.FIRST_CONTACT_ID</value>
<value>.FIRST_LOCATION_ID</value>
<value>.LOCATION_NAME</value>
<value>.GROUP_ID</value>
<value>.RESOLVER_ID</value>
<value>.RESOLVER_GROUP</value>
<value>.SOLUTION_ID</value>
<value>.ITEM</value>
<value>.COMPONENT</value>
<value>.MODULE</value>
<value>.SYSTEM</value>
<value>.TEAM</value>
<value>.USER_ID</value>
<value>.REPORTER_GROUP</value>
<value>.REPORTER_ID</value>
<value>.CURR_TARGET_DATE</value>
<value>.CURR_TARGET_TIME</value>
<value>.ACCOUNT_ID</value>
<value>.CUSTOMER_ID</value>
<value>.SOLVED_DATE</value>
<value>.SOLVED_TIME</value>
<value>.DESCRIPTION</value>
<value>.PROBLEM_RESULT</value>
</param>
<!--
=====
    Formats for parsing date strings.
=====

```

```

-->
<param name="csv.date.format.list" multivalued="yes">
<value>yyyy-MM-dd hh:mm:ss</value>
<value>yyyyMMdd</value>
<value>yyyy-MM-dd</value>
<value>yyyy/MM/dd</value>
<value>MMdd</value>
</param>
<!--
=====
    Index of a column for document id.
=====
-->
<param name="csv.column.index.id" multivalued="no">
<value>1</value>
</param>
<!--
=====
    Index of a column for date.
=====
-->
<param name="csv.column.index.date" multivalued="no">
<value>10</value>
</param>
<!--
=====
    Index of a column for document title.
=====
-->
<param name="csv.column.index.title" multivalued="no">
<value>6</value>
</param>
</data_source>
.....
.....
</ingester_config>

```

Run the Data Ingestor command to convert the CSV data into ATML based on the entries in the data_ingester_config_csv2atml.xml (Figure 10-2).

```
C:\OAE\database\IGA_SAMPLE>takmi_data_ingester.bat
conf\data_ingester_config_csv2atml.xml c:
\oae\data\raw\iga_samples.csv db\atml\IGA_SAMPLE.atml

Processing start.

Processing completed for total 88712 documents.

C:\OAE\database\IGA_SAMPLE>
```

Figure 10-2 Output of Data Ingestor command

Natural language processing

Language processing is done for all the ATML files, generated by the Data Ingestor to create output MIML files. You have to run the takmi_nlp_resource_deploy command, which allocates the necessary resources to be used in language processing, before executing the takmi_nlp command, which performs language processing on the extracted data in the ATML format. Refer to 4.6, “Natural language processing” on page 78 for more details on different steps involved in language processing.

Indexing

The OmniFind Analytics Edition preprocessor analyzes the data and extracts and records different entities of interest into MIML files. We would use the indexer component to index these MIML files for easy retrieval and search. In our case, as it is a new index, we run the takmi_index command to create the indexes. Refer to 4.7, “Indexing” on page 81 for details on the indexing process and the steps involved.

10.2.4 Basic analysis

Our focus, in this section, is more on how to analyze the results and infer meaningful insights or information from these results. OmniFind Analytics Edition provides four Web applications, Text Miner, Dictionary Editor, Rules Editor, and Alerting System, which can help you in analyzing your results. We are using mostly Text Miner and Dictionary Editor in this section to extract valuable facts and trends from the Technical Help Desk data.

To perform a basic analysis when you do not know what and where to look for information, follow these steps:

1. To start, log in to the Text miner application using the URL:

`http://localhost:9080/TAKMI_MINER`

This opens the database selection page of Text miner.

2. Select the database.

We will select IGA_HELP database, which is the database having the Technical Help Desk data, and we will be analyzing this database throughout this case study. Selecting the database takes us to the Text Miner main page which contains the category tree for the selected IGA_HELP database and the different analysis views which we are using to analyze the Technical Help Desk data. It will also have a search panel and a tools menu. Refer to Chapter 5, “Text Miner mining basics” on page 87 for detailed explanations on the Text miner user interface and features.

The category tree would have all the categories related to the Technical Help Desk data that we created while editing the category tree after exporting the data into OmniFind Analytics Edition as shown in Figure 10-3. It would also have some default categories such as *Part of Speech*, *Phrase Constituent*, and *Voice of Customer* which are the categories containing information based on the analysis done on the unstructured data. These categories would have the Technical Help Desk data classified based on the different entities extracted from the unstructured fields such as description and problem abstract (Figure 10-3).

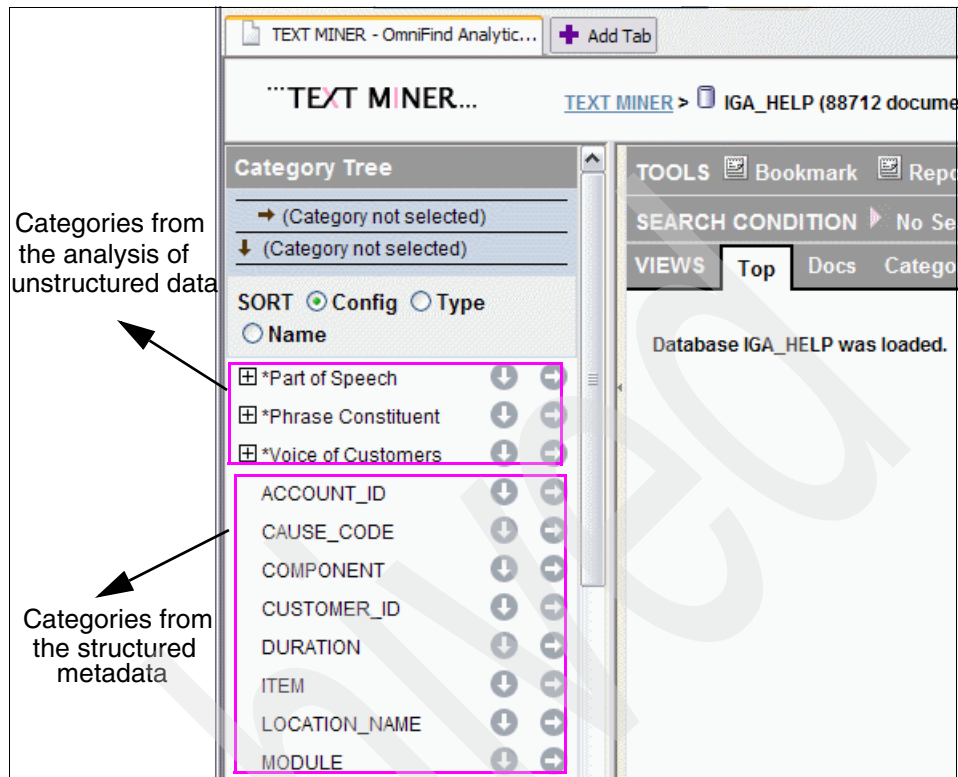


Figure 10-3 Text Miner user interface

- Now, let us try to dig into the data and see what is hidden beneath. We start by selecting *Part of Speech* as vertical category. The *Part of Speech* category contains the Technical Help Desk data categorized based on the part of speech components extracted from the unstructured data.

Click the **Category** tab to view the different keywords under this category and their frequency as shown in Figure 10-4.

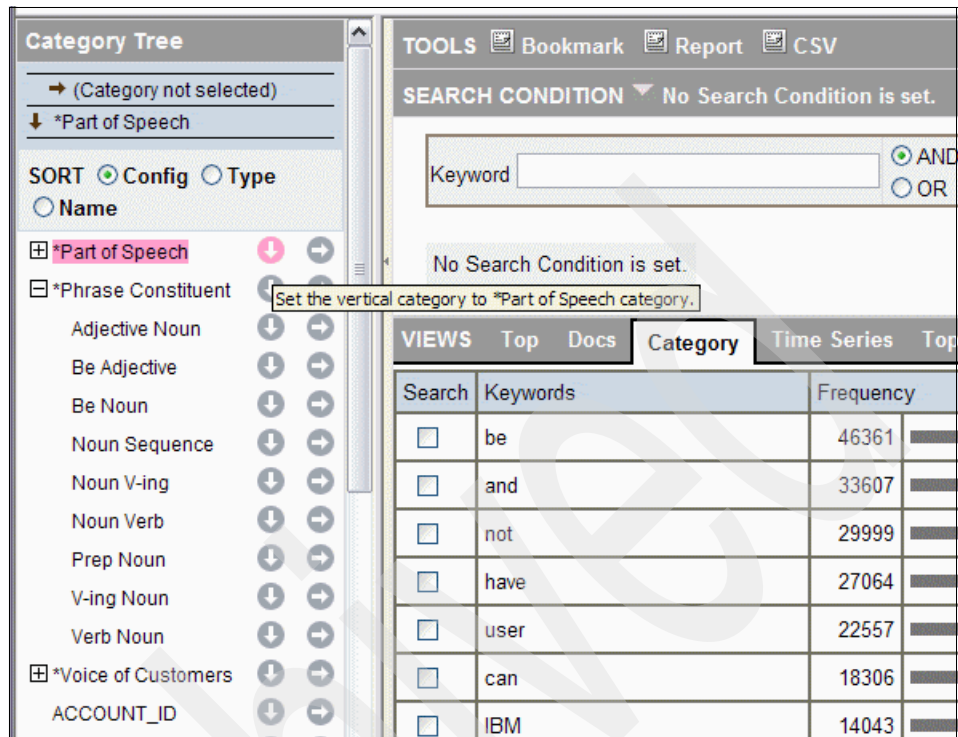


Figure 10-4 High frequency words under the Part of Speech category

We can see words such as be, and, not, have, and so on, as the mostly used words which fall under the *Part of Speech* category. These are a set of words we expect to see at a high frequency, and so this does not surprise us in anyway.

- Let us move to the next category, *Phrase constituent*. This category classifies data based on different combinations of phrases such as *Adjective Noun*, *Noun V-ing*, *Noun Verb* and so on.

Select *Phrase constituent* as the vertical category and click the category tab to view the different keywords listed under this category as shown in Figure 10-5.

Category Tree

→ (Category not selected)

↓ *Phrase Constituent

SORT ☒ Config ☐ Type

☐ Name

☒ *Part of Speech

☒ *Phrase Constituent

Adjective Noun

Be Adjective

Be Noun

Noun Sequence

Noun V-ing

Noun Verb

Prep Noun

V-ing Noun

Verb Noun

☒ *Voice of Customers

ACCOUNT_ID

Set the vertical category to *Phrase Constituent category.

TOOLS

SEARCH CONDITION ▶ No Search Condition is set.

VIEWS

↓ Category: *Phrase Constituent

List: ☐ Subcategories ☒ Keywords

Sort: ☒ Frequency ☐ Correlation ☐ Alphabet

Max lines: ☐ 20 ☐ 50 ☒ 100 ☐ 200 ☐ 500

☒ AND ☐ OR ☐ NOT

Search	Keywords	Frequency
<input type="checkbox"/>	NetView ... automation	7426
<input type="checkbox"/>	short ... name	7208
<input type="checkbox"/>	of ... record	7194
<input type="checkbox"/>	be ... first	7192
<input type="checkbox"/>	be ... recipient	7158
<input type="checkbox"/>	Lotus Notes account	7157

Figure 10-5 High frequency words under the *Phrase Constituent* category

If you look at the high frequency keywords under the *Phrase Constituent* category, you can find entries like “NetView@....automation” , “ short....name” , and so on.

- Let us drill down and check the documents under these. Select the first two keywords in the list, “NetView....automation” and “ short....name” and click **ADD a Search Condition** button as shown in Figure 10-6.

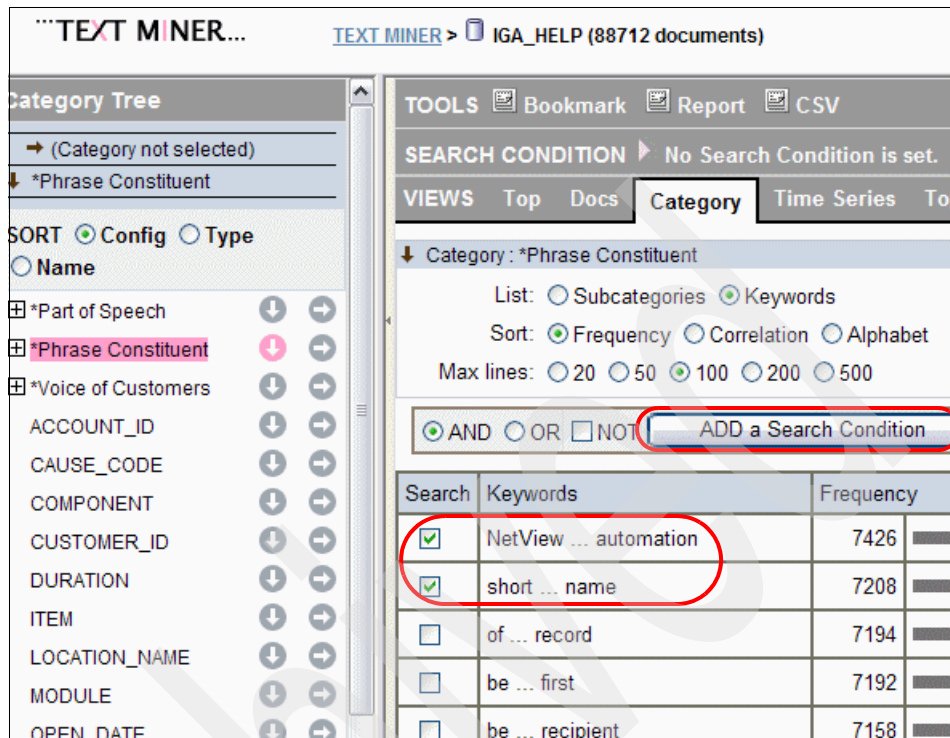


Figure 10-6 Adding a search condition

This would drill down our search from 88712 documents, which is the total number of documents loaded, to 7478 documents containing the two phrases that we selected as shown in Figure 10-7.

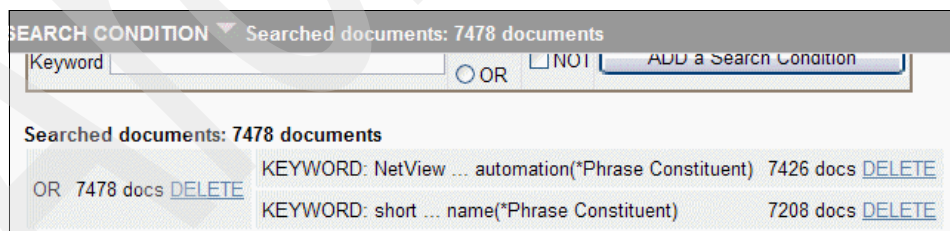


Figure 10-7 Search condition selected

- Now, let us find out what are the documents under these keywords. Click on the **Docs** tab on the View pane to view the documents containing the selected phrases (Figure 10-8).

ARCH CONDITION		Searched documents: 7478 documents					
WS	Top	Docs	Category	Time Series	Topic	Delta	2D Map
		Title	MEUC3 JOB QCH02053(JOB13303) ABENDED U065 : 20060218 08 :10				
		Text	.DESCRIPTION: MEUC3 JOB QCH02053(JOB13303) ABENDED U065 : 20060218 08 :10 08:10:16.080246-20060218 C3SSDSAERO EQQE036I JOB QCH02053(JOB13303) ENDED IN ERROR U065. PRTY=6 , 'APPL = ACHHKP@QCHREO1 , WORK STATION = C3C3, IA = 0602181900 This record has been automatically generated from the NetView automation system FGQZO running on MEUHQ If you should not have been the first recipient of this record (check the Problem Audit Trail to confirm) then you contact the Systems Automation taskid on Lotus N...				
		Detail	DETAIL				
		Standard Date	20060218				
		Title	MEUC3 JOB QCH02031(JOB13304) ABENDED U065 : 20060218 08 :10				
		Text	.DESCRIPTION: MEUC3 JOB QCH02031(JOB13304) ABENDED U065 : 20060218 08 :10 08:10:16.269645-20060218 C3SSDSAERO EQQE036I JOB QCH02031(JOB13304) ENDED IN ERROR U065. PRTY=6 , 'APPL = ACHHKP@QCHREO1 , WORK STATION = C3C3, IA = 0602181900 This record has been automatically generated from the NetView automation system FGQZO running on MEUHQ If you should not have been the first recipient of this record (check the Problem Audit Trail to confirm) then you contact the Systems Automation taskid on Lotus N...				
		Detail	DETAIL				
		Standard Date	20060218				
		Title	MEUC3 JOB QCH02151(JOB13305) ABENDED U065 : 20060218 08 :10				

Figure 10-8 Docs view for the selected search condition

When you analyze these documents, you will be able to find out that these are auto-generated messages created by the IBM Tivoli® NetView automation system, whenever the Notes servers encounter some error. At this point in time we are more interested in analyzing the complaints and issues logged by the employees and their status, rather than auto-generated messages created by the message processor systems. So let us try to group and isolate these auto-generated messages or documents, so that we can analyze the rest of the dataset without bumping into these, while searching for issues logged by the employees. If we closely examine the documents listed, we find out that all these documents have a same auto-generated format and the string “automatically generated from the NetView automation” repeats in all the documents.

Creating rules to ignore specific data entries

Let us create a new category, System message. Associate a rule with the category to detect and group all the documents containing the above string.

To create a new category, use the Dictionary Editor application as shown in Chapter 7, “Customizing the dictionary” on page 165. For details about creating a new rule, refer to Chapter 8, “Customizing rules” on page 189.

To create the new category and a rule:

1. Log in to the Dictionary Editor application using the URL:
http://yourhostname:9080/TAKMI_DIC

2. Create a category (Figure 10-9).

Click **Select database** to select the IGA_HELP database. First, we will create a category to house the auto generated documents. Click **Edit category tree** to add a new category, System messages.

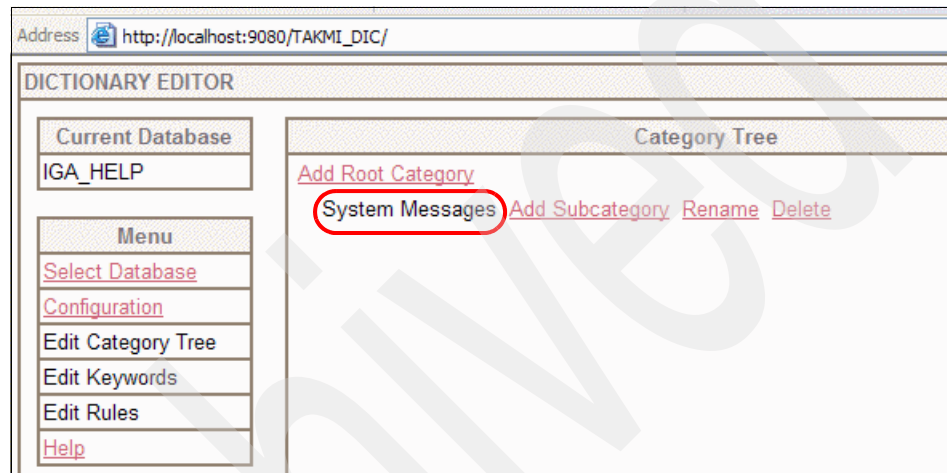


Figure 10-9 Create new category

3. Create a rule.

Once we have created the category, the next step is to create a rule that would associate all the documents having the string “automatically generated from the NetView automation” to the newly created System Messages category. Click **Edit Rules** to add a new string rule to the category, System Messages. Example 10-6 shows the pattern file for the newly created rule.

Example 10-6 Pattern file for the system messages rule

```
<?xml version="1.0" encoding="UTF-8"?>
<pattern-list lang="eng">
<mi label="-0" category=".system_messages" value="system messages">
<w id="0" str="automatically generated from the NetView
automation"/></mi>
</pattern-list>
```

You can add the rule through Rule Editor. You can also do so directly from the rule file. After creating the new category and rule, run the language processing and reindex the MIML files for the changes to be effective and to force the new category to appear in the Text Miner application. Refer to Chapter 4, “Data ingestion and indexing” on page 55 for more details on language processing and indexing.

4. Log in to the Text Miner application and check for the newly created category System Messages shows up (Figure 10-10).

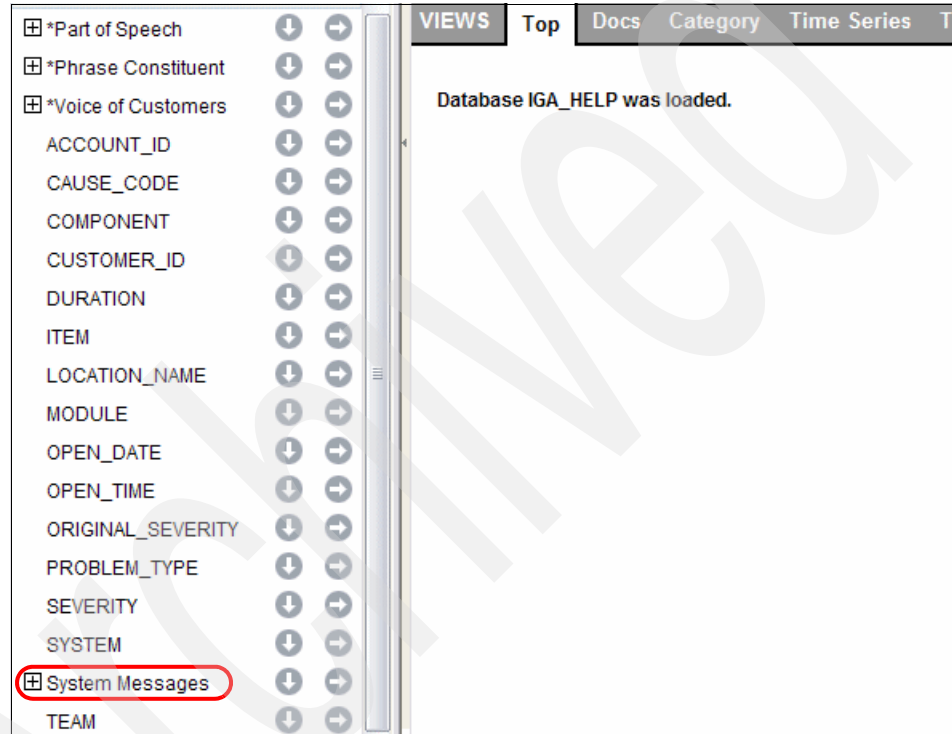


Figure 10-10 Newly created category displayed in Text Miner

5. To exclude the auto-generated documents from appearing in the analysis results, select **System Messages** as the vertical category, select the system messages keyword in the Category view, select **Not** in the Search condition and click **Add to Search Condition**. Excluding *system messages* through the search condition brings up a subset of 81178 documents from a total of 88712 documents (Figure 10-11).

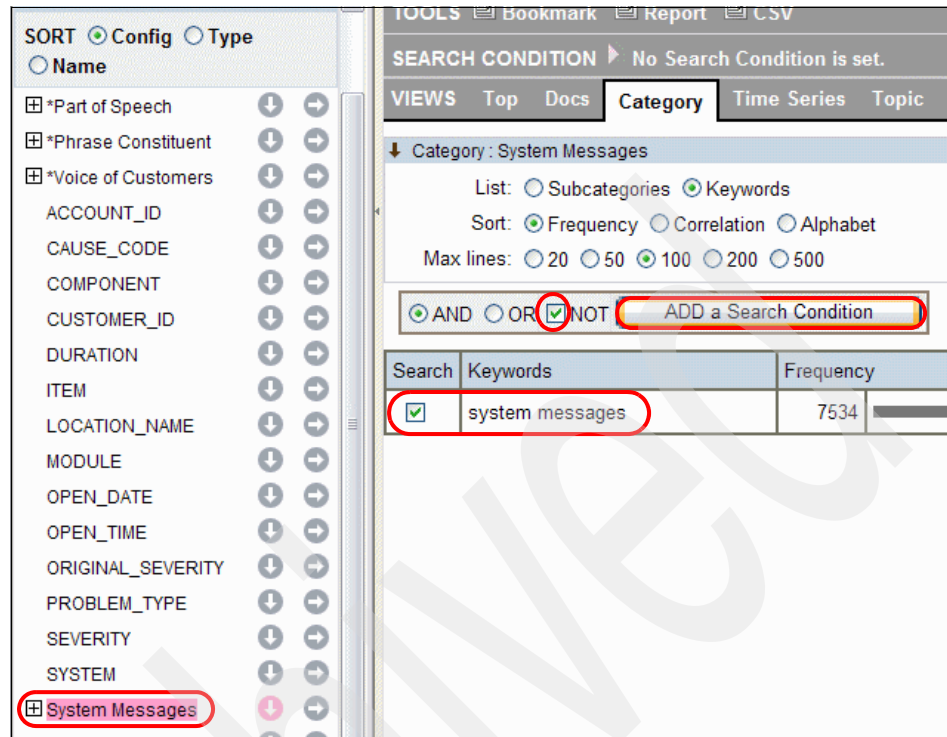


Figure 10-11 Excluding documents under category System Messages

10.2.5 Root cause analysis

We have excluded the auto generated messages from our search subset now. Let us try to analyze the problems reported by the users and drill down to the root cause of these problems to possibly identify the hidden facts which could help us in finding solutions to these problems and thereby increasing operational efficiency of the Technical Help Desk.

1. Let us start by selecting the PROBLEM_TYPE category which list all the different problems logged. Select PROBLEM_TYPE as vertical category and click the **Category** tab to view the list of problems and their frequencies.

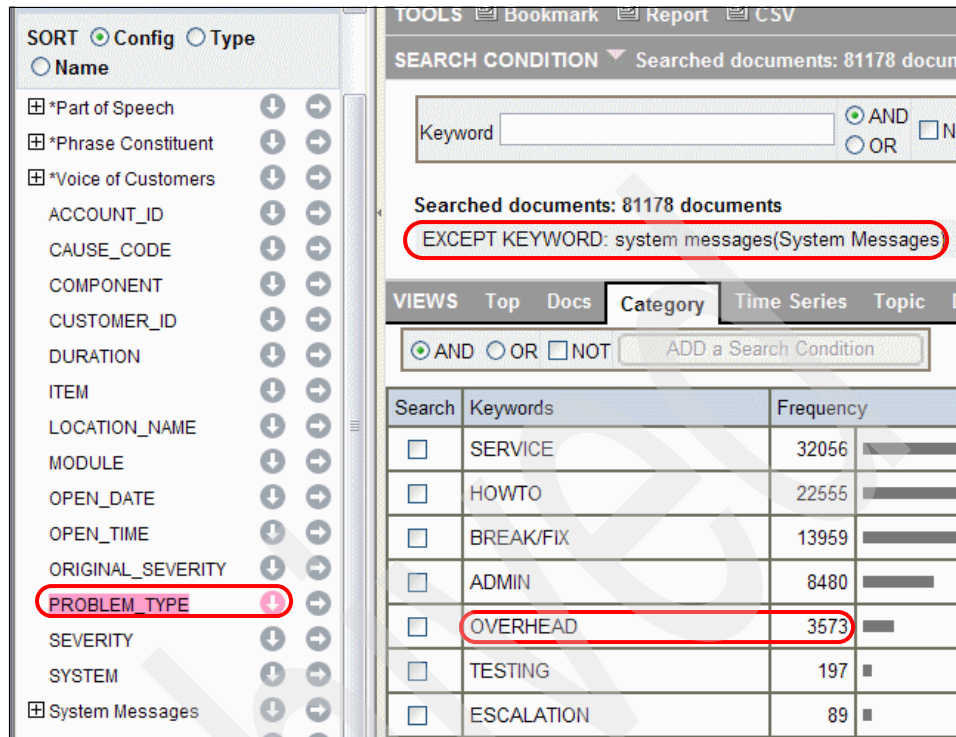


Figure 10-12 Category view for PROBLEM_TYPE category

In the category view for the PROBLEM_TYPE category, you can see keywords such as “SERVICE”, “HOWTO”, BREAK/FIX” and so on, which are predefined labels for different kinds of problems used by the Technical Help Desk analysts. Every data entry would have a problem type depending on the nature of the problem logged. As shown in Figure 10-12, the fifth category in the list is OVERHEAD, which corresponds to problems that should not have landed in the Technical Help Desk. So eliminating these calls would definitely improve the *Technical Help Desk efficiency*.

- Let us drill down and find out what exactly these problems are. Select the check box for OVERHEAD and click **ADD a Search Condition** button. This way we can narrow down to the documents pertaining to OVERHEAD category. Every time we add a search condition we are narrowing down our set of documents of interest. Adding OVERHEAD to the search condition brings up a subset of 3573 documents from a total of 88712 documents as shown in Figure 10-13.

SEARCH CONDITION ▾ Searched documents: 3573 documents

Keyword ☒ AND ☐ OR ☐ NOT [ADD a Search Condition](#)

Searched documents: 3573 documents

AND 3573 docs [DELETE](#) EXCEPT KEYWORD: system messages(System Messages) 81178 docs [DELETE](#)

KEYWORD: OVERHEAD (PROBLEM_TYPE) 3573 docs [DELETE](#)

Figure 10-13 Adding OVERHEAD search condition

Now we have narrowed down to a subset, which has user complaints that should not have landed in the Technical Help Desk.

3. We analyze this further by selecting the Voice of Customers category. Voice of Customers categorizes the documents on the basis of the user's response to the encountered problem, which can be favorable, unfavorable, question, gratitude and so on. By selecting Voice of Customers on the OVERHEAD subset, we can identify the major issues or responses logged by the users. To view these, select Voice of Customers as the vertical category and view the category tab (Figure 10-14).

Category Tree

→ (Category not selected)

↓ *Voice of Customers

SORT ☒ Config ☐ Type

☐ Name

☒ *Part of Speech

☒ *Phrase Constituent

☒ *Voice of Customers

☒ Favorable

☒ Gratitude

☒ Question

☒ Requirement

☒ Unfavorable

ACCOUNT_ID

CAUSE_CODE

COMPONENT

CUSTOMER_ID

DURATION

ITEM

LOCATION_NAME

MODULE

TOOLS [Bookmark](#) [Report](#) [CSV](#)

SEARCH CONDITION ▾ Searched documents: 3573 documents

Keyword ☒ AND ☐ OR ☐ NOT [ADD a Search Condition](#)

Searched documents: 3573 documents

AND 3573 docs [DELETE](#) EXCEPT KEYWORD: system messages(System Messages) 81178 docs [DELETE](#)

KEYWORD: OVERHEAD (PROBLEM_TYPE) 3573 docs [DELETE](#)

VIEWS [Top](#) [Docs](#) [Category](#) [Time Series](#) [Topic](#) [Delta](#) [2D Map](#)

		Time Series	Topic	Delta	2D Map
<input type="checkbox"/>	unable to find	174		0.3	
<input type="checkbox"/>	good	174		0.4	
<input type="checkbox"/>	want to reset	168		0.5	
<input checked="" type="checkbox"/>	need telephone	166		12.4	
<input type="checkbox"/>	unable to get	166		0.4	
<input type="checkbox"/>	want to install	158		1.2	
<input type="checkbox"/>	thank	143		0.2	
<input type="checkbox"/>	want support	143		3.6	
<input type="checkbox"/>	unable to log	135		0.5	
<input type="checkbox"/>	want ... help with	132		4.4	

Figure 10-14 Category view for Voice of Customers category

In the category tab in Figure 10-14 on page 265, you can see keywords such as “unable to find”, “good”, and “want to reset”, listed in the descending order of their frequencies. The fourth entry in the list, “need telephone” has a higher than normal correlation with the OVERHEAD subset.

- Let us drill down and get to the root cause of this. Select the “need telephone” keyword and add it to the search condition. This brings up a subset of 166 documents which is common to both OVERHEAD and need telephone subsets (Figure 10-15).

Searched documents:166 documents	
	EXCEPT KEYWORD: system messages(System Messages) 81178 docs DELETE
AND 166 docs DELETE	KEYWORD: OVERHEAD (PROBLEM_TYPE) 3573 docs DELETE
	KEYWORD: need telephone(*Voice of Customers) 166 docs DELETE
VIEWS	Top Docs Category Time Series Topic Delta 2D Map
Title	boulder caller needs telephone support
Text	.DESCRIPTION: boulder caller needs telephone support. .PROBLEM_RESULT: x-fer call to tasha from sbc.
Detail	DETAIL
Standard Date	20060103
Title	t/l 295-7801 need telephone support in Poughkeepsie
Text	.DESCRIPTION: Caller: [redacted] Tiv. 26523604 Site: Poughkeepsie Problem need telephone support.//gn .PROBLEM_RESULT: Tiv. 26523604 TSA action: Transferred to Carol @ SBC.//gn
Detail	DETAIL
Standard Date	20060104

Figure 10-15 Docs view for the need telephone-OVERHEAD subset

As we look through this list of documents, there are many cases where the calls are coming into Technical Help Desk which should have been routed to the Telephone Help Desk (that resolves telephone related issues). This needs to be fixed upstream by routing the calls correctly. One way of achieving this can be by enhancing the menu selection that users go through on the IVR with explicit messages about calling the Telephone Help Desk for telephone related issues. Reducing these calls would result in significant decrease in operational costs.

In very few steps, we demonstrated the usage of OmniFind Analytics Edition to analyze 88712 documents to identify calls that the support center should not handle. By identifying this issue and resolving it, the improvement to the system would increase productivity.

Other types of problems discovered through this type of analysis include problems caused by specific parts of a computer. After much investigation, it was discovered later that the computer component is not manufactured to standard. With redesign of the specific parts, the problem is solved. Support center handling of this type of issues has been reduced.

10.3 Case study: NHTSA

This case study applies analysis to sample data that is downloaded from the National Highway Traffic Safety Administration (NHTSA). The case study details the usage of IBM OmniFind Analytics Edition to analyze the complaints registered by the state traffic police and vehicle drivers to discover common concerns related to motor vehicles and safety.

We showcase search condition buildup, 2D Map view, and mining information using Part of speech with Category view to look for high frequency and high correlation keywords. In addition, we present the need for dictionary customization and how it can be done.

10.3.1 Overview of the business scenario

NHTSA is a government agency in United States of America whose mission is to save lives, prevent injuries, identify vehicle related crashes, and reduce them.

To accomplish their mission, here are a few practices adopted by the agency:

- ▶ Conducts controlled crash tests and publishes results on new models from all the car manufacturers.
- ▶ Publishes recall information about equipment found to be faulty by the manufacturers and dealers.
- ▶ Provides a Web interface to allow traffic police and vehicle drivers to register complaints on accidents and notable vehicular incidents.
- ▶ Publishes product ratings on car seats from various manufacturers.
- ▶ Publishes databases that relate to complaints and product recalls and crash investigations for free downloads.

Refer to the NHTSA Web site at <http://www.nhtsa.gov/> to learn more about the agency.

The agency wants to be able to identify any malfunctioning or mal-manufacturing components in a particular vehicle model or a particular model that is causing abnormally high number of incidents. The agency needs to discover any unusual problems related to auto company makes, models, components, location, and other factors, and investigate and discover root causes or trend for particular type of incidents.

Agency goal: Identify potential health and safety hazard on the road through complaints logged on the Web site. Take preventive measure to avoid future accidents caused by the hazard.

10.3.2 NHTSA database details

For the scenario described in this case study, we download the NHTSA complaints database and apply analytics to the data.

The NHTSA complaints database is available to download at:
<http://www-odi.nhtsa.dot.gov/downloads/list.html>.

The NHTSA complaints data contains all safety-related defect complaints received by NHTSA since January 1, 1995.

Note: NHTSA updates the published data on a periodic basic. The data might be changed since we last downloaded the data.

The NHTSA complaints database consist of multiple rows of complaint incidents. Each row contains multiple column data separated by a TAB character. The NHTSA complaint database resembles contents of a relational database table.

Table 10-2 outlines the schema of the NHTSA database.

Table 10-2 Schema for the NHTSA complaints database

S No	Column Name	Data Type and description
1	CMPLID	CHAR(9) NHTSA's Internal Unique sequence number.
2	ODINO	CHAR(9) NHTSA's Internal reference number. This number can be repeated for multiple components.

S No	Column Name	Data Type and description
3	MFR_NAME	CHAR(40) Manufacturer's name
4	MAKE	CHAR(25) Vehicle/equipment make
5	MODEL	CHAR(256) Vehicle/equipment model
6	YEAR	CHAR(4) Model year, 9999 if unknown or N/A
7	CRASH	CHAR(1) Was vehicle involved in a crash, 'Y' or 'N'
8	FAILDATE	CHAR(8) Date of incident (YYYYMMDD)
9	FIRE	CHAR(1) Was vehicle involved in a fire 'Y' or 'N'
10	INJURED	NUMBER(2) Number of persons injured
11	DEATHS	NUMBER(2) Number of fatalities
12	COMPDESC	CHAR(128) Specific component's description
13	CITY	CHAR(30) Consumer's city
14	STATE	CHAR(2) Consumer's state
15	VIN	CHAR(11) Vehicle's VIN#
16	ADATE	CHAR(8) Date added to file (YYYYMMDD)
17	LDATE	CHAR(8) Date complaint received by NHTSA (YYYYMMDD)
18	MILES	NUMBER(6) Vehicle mileage at failure
19	OCCURENCES	NUMBER(4) Number of occurrences

S No	Column Name	Data Type and description
20	CDESCR	CHAR(2048) Description of the complaint. Free text data that represents a complaint. This free text description will be broken down into nouns, adjectives, verbs or other parts of speech by the language processing annotators in OmniFind Analytics Edition. For the rest of the case study, we identify the contents of this column as unstructured source of data.

From Table 10-2, the bold field, CDESCR, is an unstructured data field.

As a prerequisite to use OmniFind Analytics Edition's batch commands, we followed the instructions provided by NHTSA to import the downloaded data and convert it into CSV format. Refer to the import instructions provided by NHTSA at http://www-odi.nhtsa.dot.gov/downloads/folders/Complaints/Import_Instructions.pdf.

With the NHTSA complaints database now available in the desired CSV format, you can refer to Chapter 4, "Data ingestion and indexing" on page 55 to learn about the building a OmniFind Analytics Edition index.

10.3.3 Basic analysis

In this scenario, we are concerned with identifying components and the associated vehicle model that were reported in the most number of *fire*-related incidents. Once we identify the component and the model, we can read the reported complaints and understand the component problem.

Note: Vehicle manufacturer names and models are blacked out.

In all the figures to follow, we have to black out the names of vehicle manufacturers and models.

Customizing dictionary (creating a new category)

Because we want to identify components of vehicles that cause fire-related incidents, we therefore want to mine any vehicle component related information in the complaints. To do so, we customize the OAE dictionary to list all vehicle component related words in a new category called Component Description.

Using the Dictionary Editor, we add the components-related keywords in the Component Description category. Keyword examples are:

alarm
air bag (or airbag)
control module
seal
parking light
spare tire
fuel injection
wheel
heating system
timing belt
engine
engine check light
cruise control
cruise control switch

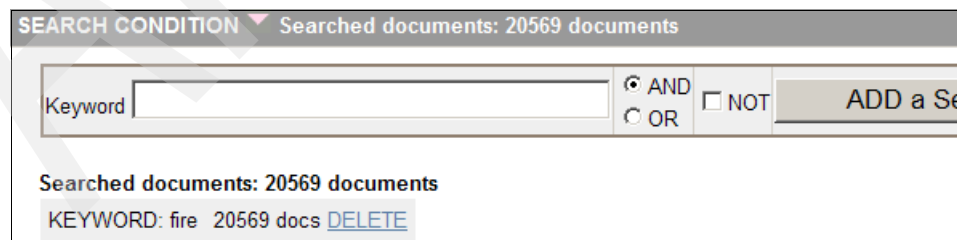
Model of cars in the complaints are provided (in the structured content field). Therefore, we do not need to do any special customization to mine data associated with a particular model. This scenario is an example where we can use both the structured content with unstructured content to analyze data and get information that we are looking for.

Using both search and 2D Map (on Model and Component Description), we can narrow down the components that are associated with fire incidents.

Fire incident analysis

The following are the sequence of actions we use to identify problem components and models that are associated with fire incidents:

1. Enter *fire* in a keyword search. Figure 10-16 shows the results of the keyword search. We discover there are 20569 incidents out of the 501376 complaints registered in the NHTSA database.



The screenshot shows a search interface with a header bar that reads "SEARCH CONDITION" and "Searched documents: 20569 documents". Below this is a search form with a "Keyword" input field, a radio button for "AND", a radio button for "OR", and a checkbox for "NOT". To the right of the "NOT" checkbox is a button labeled "ADD a Se". Below the search form, the text "Searched documents: 20569 documents" is displayed, followed by "KEYWORD: fire 20569 docs" and a "DELETE" link.

Figure 10-16 Search condition for fire

2. We are interested in specific components and their associated models that are related to fire. So, from the Category tree, we select **MODEL** using the horizontal arrow and **COMPONENT DESCRIPTION** (vertical arrow). From the VIEWS, we select the **2D Map** view. Figure 10-17 shows the search results.

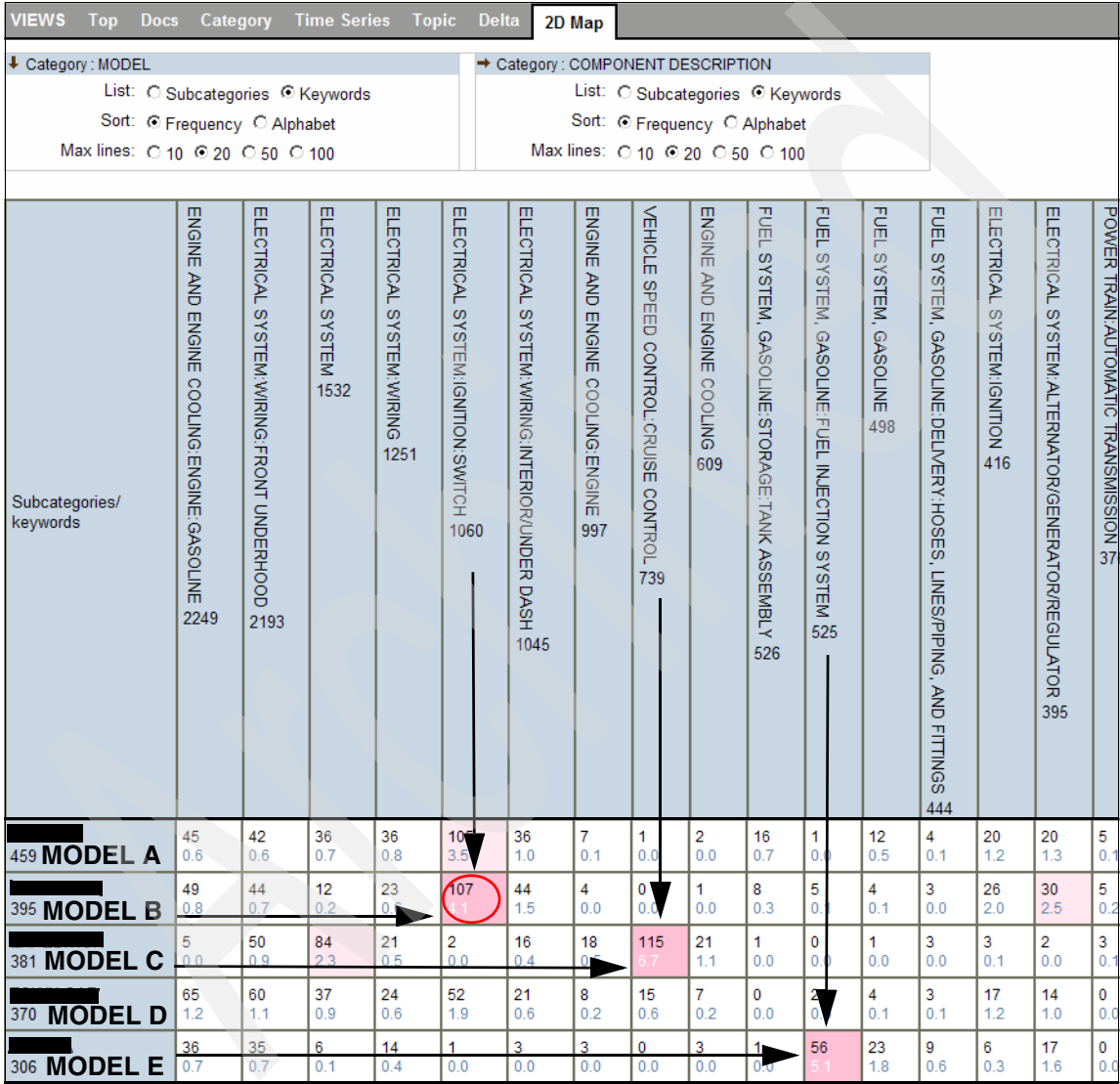


Figure 10-17 2D map with Car models and categories on a fire search condition

From Figure 10-17, we can easily discover high frequency and correlations between the following components and their associated models:

- ELECTRICAL SYSTEM:IGNITION:SWITCH and MODEL B
- VEHICLE SPEED CONTROL:CRUISE CONTROL and MODEL C
- FUEL SYSTEM, GASOLINE:FUEL INJECTION SYSTEM and MODEL E

10.3.4 Root cause analysis

After we know the components with the associated vehicle models that cause fire accidents, let us now focus on discovering the root cause of problems. That is, let us see why the identified components are causing fire.

For our scenario, we focus on the root cause of ELECTRICAL SYSTEM:IGNITION:SWITCH and MODEL B.

Continuing from the previous analysis, we discover the root cause of the problem as follows:

1. Click the circle highlighted in Figure 10-17. This action generates a new search condition to narrow down a database with 501376 documents to 107. Figure 10-18 shows the resulting search condition.

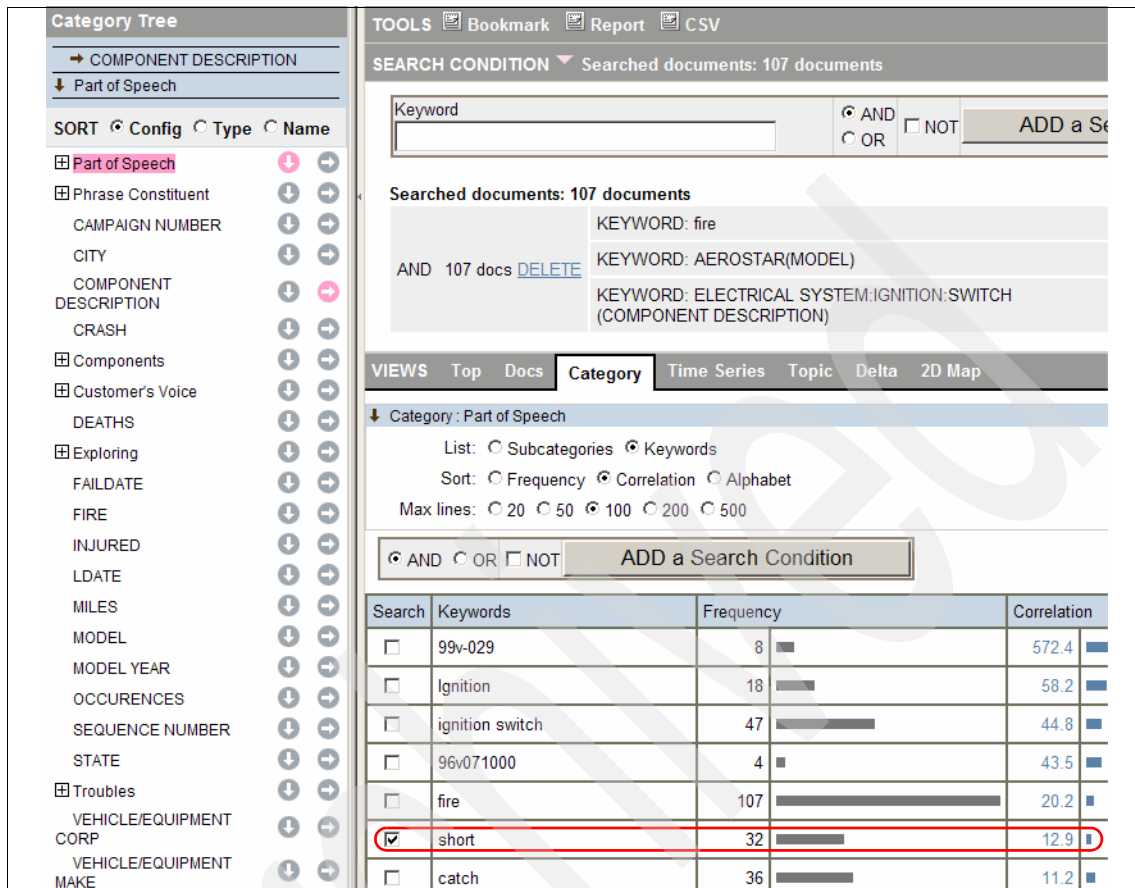


Figure 10-18 listing by POS category

- We now narrow down the actual problem with ignition switch. The *fire* could be triggered by over heating of the switch or wires get short circuit. This can be done in different ways. In Figure 10-18, we display the 107 documents categorized by the *Part of Speech*. When the list is sorted by *correlation*, we discover the keyword *short* is referred in 32 documents out of 107 documents and has a correlation of 12.9.
- Because *short* has a high correlation, we drill down on the related documents. Select the keyword *short* and add it to the *search condition*. Figure 10-19 shows the document view of the resulting 32 documents.

Part of Speech

Config Type Name

Part of Speech

Phrase Constituent

CAMPAIGN NUMBER

ITY

COMPONENT DESCRIPTION

RASH

Components

Customer's Voice

FEATHS

Exploring

UPDATE

RE

JURED

DATE

LES

ODEL

ODEL YEAR

CCURENCES

SEQUENCE NUMBER

ATE

oubles

CHICLE/EQUIPMENT

P

CHICLE/EQUIPMENT

E

N

ice of Customers

Keyword

AND OR NOT

ADD a Search Condition

Searched documents: 32 documents

KEYWORD: fire 20569 docs [DELE](#)

KEYWORD: AEROSTAR(MODEL) 2564 docs [DELE](#)

AND 32 docs [DELETE](#) KEYWORD: ELECTRICAL SYSTEM:IGNITION:SWITCH (COMPONENT DESCRIPTION) 5344 docs [DELE](#)

KEYWORD: short(Part of Speech) 7502 docs [DELE](#)

VIEWS Top Docs Category Time Series Topic Delta 2D Map

Documents

Docs per page: 10 20 50 100

<< 1 - 20 >>

Title	128026
Text	DESCRIPTION: Electrical short in the ignition switch, causing the vehicle to catch on fire in the steering column (the vehicle was in the off position). *ak
Detail	DETAIL
Standard Date	19970123

Title	46441
Text	DESCRIPTION: Vehicle was parked, engine compartment fire , under dash and steering column, vehicle complete burned only the front, electrical short . *ak
Detail	DETAIL
Standard Date	19960419

Title	29485
Text	DESCRIPTION: Vehicle ignited in fire due to ignition switch short . Ti

Figure 10-19 Docs view of the search condition

- Click **DETAIL** on one of the document to see the complaint description logged by people. Figure 10-20 shows the details of one of the listed documents. From the complaint description, we find *the ignition switch caused fire because of short circuit*. The other complaints in the listed documents can be read to confirm the similarity.

Document Information	
Document ID	128026
Title	128026
Text	DESCRIPTION: Electrical short in the ignition switch, causing the vehicle to catch on fire in the steering column (the vehicle was in the off position). *ak
Standard Information	
Category	Keyword
CAMPAIGN NUMBER	808029
CITY	HIA LEAH
COMPONENT DESCRIPTION	ELECTRICAL SYSTEM:IGNITION:SWITCH
CRASH	N
FAILDATE	19961029
FIRE	Y
MODEL	MODEL B

Figure 10-20 Details of a listed document.

- The information discovered by the TAKMI MINER can be leveraged by NHTSA to contact the manufacturer and get the product recalled and fixed.

In very few steps, we demonstrated the usage of OmniFind Analytics Edition to analyze 501376 documents to discover that short circuit of ignition switches in *MODEL B* is the cause for quite a number of fire related incidents. To achieve the similar findings by humans, it could take considerable resources, time and energy.

10.3.5 Sample view usage

As discussed in Chapter 5, “Text Miner mining basics” on page 87, OmniFind Analytics Edition provides many views that help you analyze your data. We reiterate some of the views we covered in that chapter here for the completeness of the case study.

Time Series view

Using the Time Series view, you can see the frequency count of the search results over a period of time.

Figure 10-21 shows there are 94 occurrences of fire related incidents by each year. From the view, we discover an abnormal increase in the fire related incidents in year 1995 (increase from 10 in year 1994 to 50 in 1995). This view is

ideal for analysts to identify abnormal trends or study the impact of some positive fix in automobiles. A sudden increase in the frequency during a particular period of time usually requires further investigation. This might be where you can discover a bad auto component produced in a certain year that is causing a lot of problems.

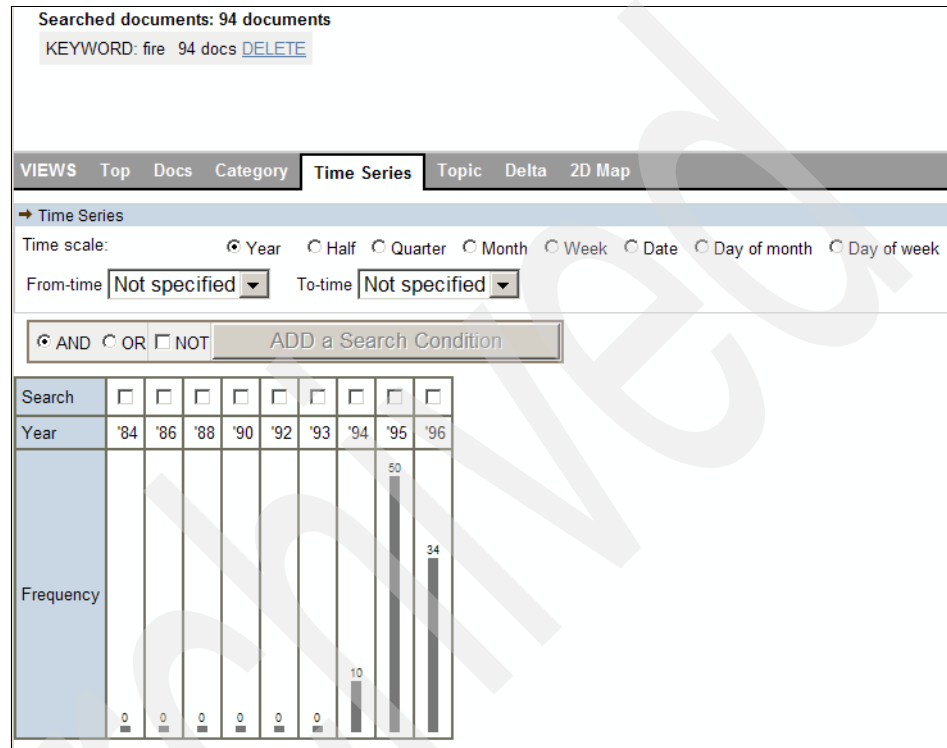


Figure 10-21 Time Series view

Topic view

In a Topic view, the frequency counts and deviations of the keywords are displayed over a period of time. Unexpected spikes in the deviations are marked by pink bars. The highlighted spikes are hints for the analysts for more in-depth analysis.

Figure 10-22 shows 20569 incidents that have occurrences of the keyword *fire*. Some cells are highlighted, one of which is 675 and 90. The numbers indicated that there were 675 incidents reported with the keywords *fire* and *ELECTRICAL SYSTEM:IGNITION:SWITCH* in 1996 and 90 incidents in 1997. These incidents are above the average frequency counts when computed over 1984 to 1997. To discover the root cause of the issues, the analyst can select the highlighted cell,

add that to additional search, and investigate any unusual problems associated with ignition switch and fire.

This view is ideal for analysts to quickly identify abnormal trend and perform further investigation. Any frequency count that deviates (increases or decreases) from the computed average years are highlighted and you can further drill down.

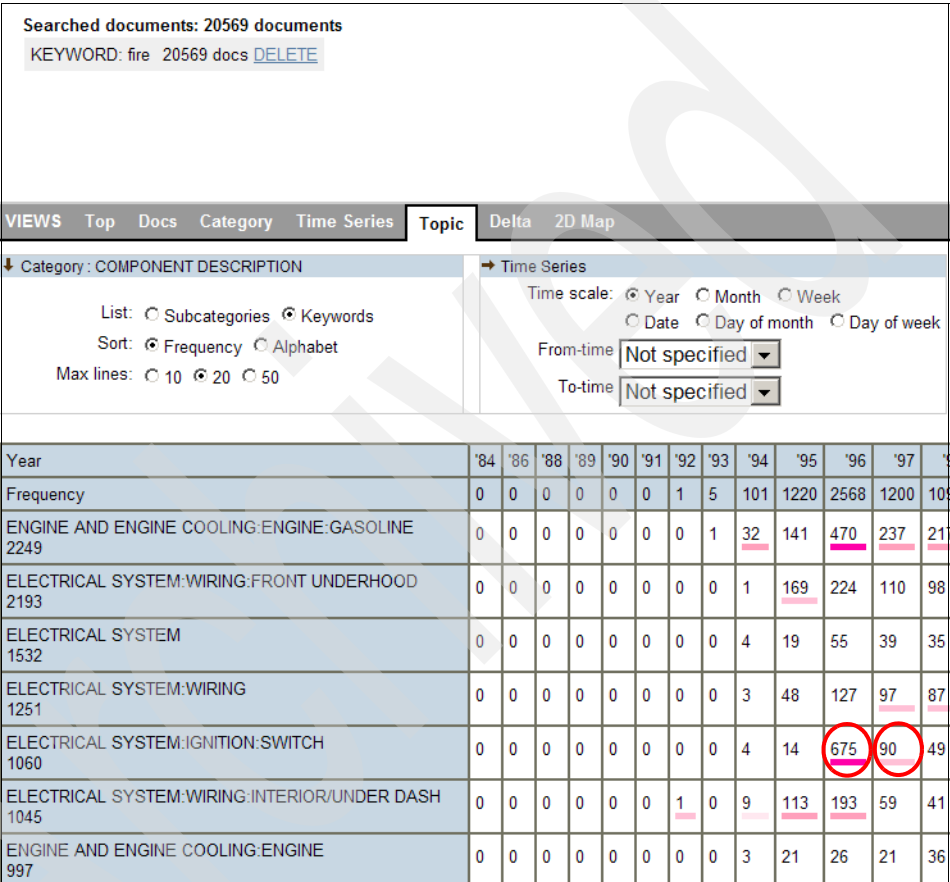


Figure 10-22 Topic view

Delta view

In the Delta view, the keyword frequencies and increases indicators are displayed against a time scale. The increase indicators refer to the increase in the frequencies of keywords and subcategories along a selected category. This is achieved by measuring how much the frequency on each given date deviates from a constant frequency.

Figure 10-23 shows Delta view of keywords associated to the COMPONENT DESCRIPTION category found across 20569 documents that contain the word fire. Notice two highlighted numbers. The first one, 46.4 indicates a high increase in frequency for ELECTRICAL SYSTEM:WIRING:FRONT UNDERHOOD associated keywords from years 1995 to up. This hints the analyst a problem has been detected with wiring, front underhood in these years and the root cause might be associated with fire incidents.

This type of increases is an indication to analysts as an early warning of anomaly. Further investigation should be done to drill down the root cause of the problem.

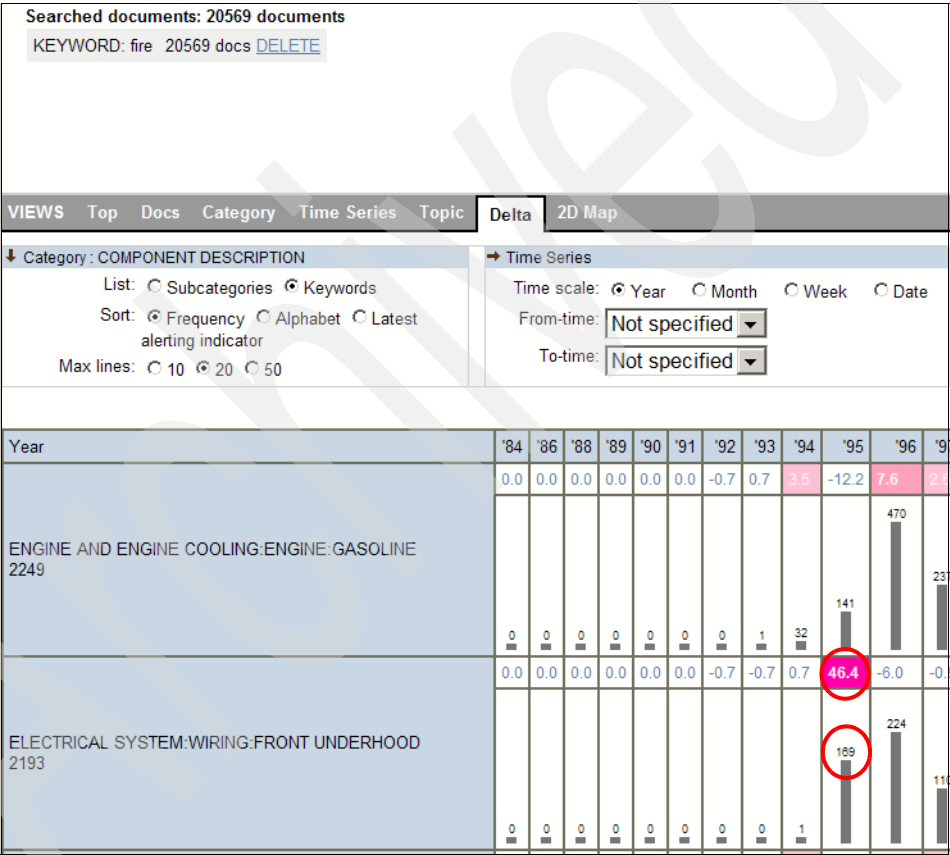


Figure 10-23 Delta view

10.4 Case study: e-mail database

This case study applies text analysis to e-mail data. The dataset we used was taken from the correspondence of the team who wrote this book. The case study details the usage of IBM OmniFind Analytics Edition to analyze the e-mails to discover potential problems and concerns related to a business process.

We showcase the flexibility and power of the product through dictionary and rules customization.

10.4.1 Overview of the business scenario

Every company has e-mail databases that store e-mails between its employees and with external sources. The type of data with which we are dealing in this use case is a generic e-mail database that everyone uses.

With every e-mail exchange in corporate realms, there will be a certain compliance policy. Such a policy typically contains provisions for legal liability, breach of confidentiality, and damage to corporate reputation:

- Legal liability:

Electronic data of all types is now admissible as evidence in legal proceedings. The variety of cases, the types of data, and the nature of the legal queries have made a text analytics system, which can discover relevant content in unstructured sources, indispensable.

- Breach of confidentiality:

Employees must be careful what they send out in their e-mails. Employees should familiarize themselves with what their employer considers confidential information, which differs from company to company. While the employee might be expected to know better than to send out new product data, this kind of information is sometimes fairly subtle. Another common oversight is simply talking about employees to other employees. An analytics surveillance and compliance system might find cases of breaches of confidentiality.

- Damage to reputation:

There are any number of ways employees can intentionally or inadvertently damage their company's reputation via e-mail correspondence. They include foul language and references to drugs — topics that no one would probably talk about in public; however, some people tend to think their e-mail correspondence are private, despite the fact that they are using their corporate e-mail accounts. On the other end of the spectrum, employees should not slander their employers directly or indirectly. To discover this kind of information requires a large number of complex analytics.

It is also a concern of a company to investigate if there are any problems or issues related to a particular business practice, process, organization, and project. This is what we focus on in this case study.

Scenario goal: Discover any employee’s problems and issues that are related to a particular project, work process, or business practice. Specifically, we want to discover problems and issues related to Visa applications through analysis of e-mail databases.

10.4.2 e-mail database details

We obtain some Notes databases (.nsf) that contain e-mails from people recently involved in foreign travel. Using OmniFind Enterprise Edition, e-mail are pulled from the databases and parsed for OmniFind Analytics Edition. See details in Chapter 11, “Integrating OmniFind Analytics Edition with OmniFind Enterprise Edition” on page 295.

All work in the database is done from a subdirectory of OmniFind Enterprise (Figure 10-24).

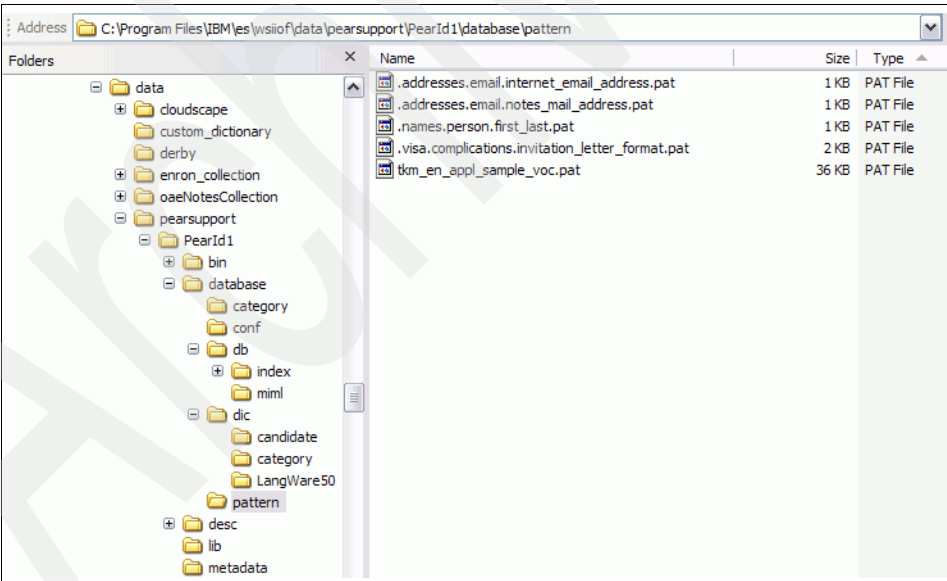


Figure 10-24 Typical directory structure for an OmniFind Analytics Edition pear database

10.4.3 Basic analysis

We begin by making generalizations about Visa application problems and issues. It is obvious that we should group associated problems and issues under a new category, Visa.

New category:

- ▶ Visa

What kinds of topics are usually of concern with Visa applications? People from different countries have different Visa requirements. If these requirements are not met, they would be prevented from coming to the country (in this case, Japan). Our hypothesis gets translated into a subcategory in the category tree called Requirements.

New category with its subcategory:

- ▶ Visa
 - Requirements.

What words that appear in e-mails that should fall into the Visa Requirements category? For this, we have to come up with a good list of keywords associated with Visa Requirements. It is a good idea to do some search from online to develop a common list of words that might fall into the Visa requirement process. We found this set of requirements on an IBM travel page:

Obtaining new passport

1. You can obtain a new passport from your home country Embassy in Tokyo (confirm requirements with the Embassy). For U.S. assignees, documents necessary are your present passport and 3 passport size pictures (2x2"). The application form is available at the Embassy.

It is necessary to register the change of passport number at your local ward (ku) office. They will change the passport number in your alien registration card.

1. Visa

a. Japanese visa

Visa transfer from an old passport to a new one is no longer available since early January 1994. Therefore, it is recommended to carry old and new passports if you have obtained a new passport.

b. Visas of countries other than Japan Contact the appropriate embassy and follow its instructions on visa transfer.

2. Transferring "Multiple Re-Entry Permit" and "Period of Stay"
Immigration Proxy can handle the transfer on behalf of assignees
residing in Tokyo.

To transfer "Multiple Re-Entry Permit" and "Period of Stay" from the
old passport to the new passport, submit your completed form <Petition
for Transfer of Endorsement> and both old and new passports to your
Japan IAR by Friday noon for Immigration proxy. Your new passport
bearing transferred multiple re-entry permit and period of stay as well
as your old passport will usually be
ready for pick-up at your Japan IAR office on or by following
Wednesday.

If you need to have one transferred urgently or you are residing in a
location other than Tokyo, please show up at the Regional Immigration
office in person with old and new passports. Form is available at the
Immigration office.

Passports / Japanese Visas

An outside vendor assists with initial procedures; Thereafter, assignees
are expected to pay personally if they wish to have assistance.

Derived from the above description (which you can also do by sampling some
e-mails that are requirements related), we come up with the following keywords
to be part of the Visa Requirements category:

visa
requirement
passport
application
embassy
ward
multiple
re-entry
permit
period
stay
immigration
proxy
petition
transfer
endorsement
IAR
regional
pictures
number

These must be imported into the Dictionary Editor as candidates and associated with the subcategory Requirements. A couple of terms require synonyms (for details on importing candidate lists, see 7.3, “Working with keywords and synonyms” on page 176). Figure 10-25 shows the completed import.

Display Mode Category Tree

RESET CATEGORY SEARCH

Category Tree

- Addresses ADD
- Names ADD
- Visa ADD
 - Complications ADD
 - Requirements ADD

Registered Keyword List (20 keywords)

<< 1 - 20 >>

SELECT ALL NEW KEYWORD

Select	POS	Keyword	Synonym	Category
<input type="checkbox"/>	Noun	re-entry =multiple re-entry	EDIT	Requirements REMOVE
<input type="checkbox"/>	Noun	immigration	EDIT	Requirements REMOVE
<input type="checkbox"/>	Noun	petition	EDIT	Requirements REMOVE
<input type="checkbox"/>	Noun	period =period of stay	EDIT	Requirements REMOVE
<input type="checkbox"/>	Noun	ward =ku	EDIT	Requirements REMOVE
<input type="checkbox"/>	Noun	endorsement	EDIT	Requirements REMOVE
<input type="checkbox"/>	Noun	application =visa application	EDIT	Requirements REMOVE

Figure 10-25 Registering Visa Requirements keywords in the Dictionary Editor

With the direct method of knowledge discovery, we should now be able to just apply this set of keywords to the dictionary, and see what kind of associations OmniFind Analytics Edition comes back with. Run natural language preprocessing and indexing on our database and let us see what we got.

By using 2D Map view on e-mail sender and Requirements subcategory, we can quickly see any high correlation between senders and particular Requirements keywords. See Figure 10-26.

Vertical category: Requirements List: Keywords Sort: Frequency Max lines to display: 20				Horizontal category: First and Last List: Keywords Sort: Frequency Max lines to display: 20				
Subcategories/keywords	Kameron Cole 222	Srinivas Chittivelli 189	Ratheesh Muralidharan 97	Scott Harms 88	Deanna Polin 80	Iwao Inagaki 56	Tetsuya Nasukawa 32	Joel Waterman 20
number 71	37 0.6	45 0.8	12 0.3	8 0.1	42 1.8	17 0.8	2 0.0	0 0.0
application 46	24 0.5	29 0.7	13 0.5	6 0.1	15 0.7	19 1.5	10 1.0	0 0.0
period 40	17 0.4	26 0.7	3 0.0	0 0.0	31 2.3	5 0.2	0 0.0	0 0.0
requirement 32	11 0.2	21 0.7	0 0.0	0 0.0	19 1.5	19 2.1	9 1.3	6 0.9
passport 32	4 0.0	28 1.1	0 0.0	0 0.0	32 3.0	15 1.5	0 0.0	0 0.0
stay 15	0 0.0	15 1.0	0 0.0	0 0.0	15 2.5	15 3.5	0 0.0	0 0.0

Figure 10-26 What trends appear?

The dark numbers in the squares show how many e-mails appear in both the sender and Requirements category. The fraction indicates the correlation between the two. Notice the high correlation between one person and period, passport, and stay. Another high correlation occurs between another person and requirement and stay.

Since the two people in question are in charge of the business process related to Visa application for the company, we can exclude them from the list. Otherwise, in general condition, you might want to investigate to see why there is a high correlation between particular people and keywords associated to a business process, project, practice, or other category under analysis or investigation.

After excluding them from the results, we see other names appear at the top of the correlation list. See Figure 10-27.

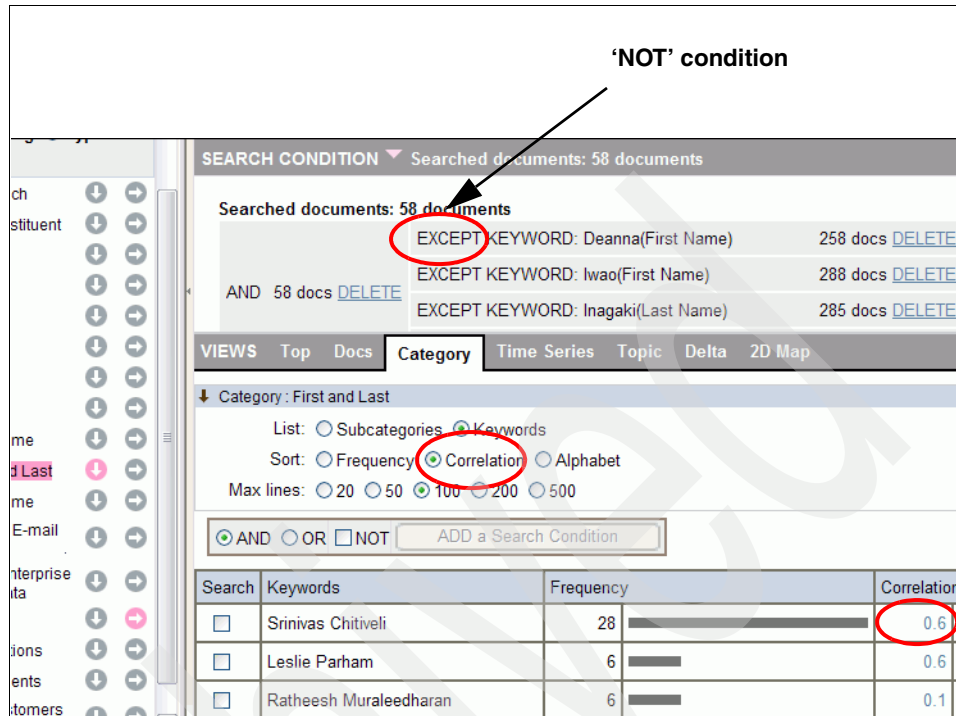


Figure 10-27 We have a winner!

The result is sorted by Correlation using the Category view. Notice that the correlation is less than 1, which is normal. In general cases, you might see a high correlation indicating that some people in your company might have specific problems and issues with the business process, project, or practice. You can drill down to see the root cause of the problems.

Ultimately, just using the list of terms for the Dictionary is not sufficient for the Visa Requirements category. As it stands now, we are getting other requirements that do not pertain directly to Visa applications. Also, we are missing text such as this:

Please see his note below regarding VISA requirements. He will require an invitation letter from IBM Japan for VISA processing.

We need to take into account the word VISA spelled using all capitals. We could, of course, enter VISA as a synonym; however, for the sake of example, and to gain more power for our category, we configure it by adding this rule to the category:

```
<mi category=".visa.requirements" value=" ${0.lex} ${1.lex}">
<w id="0" pos="noun" lex="/^(visa)|(VISA)$/">
<w id="1" pos="noun"/></mi>
```

This produces these results in Text Miner. See Figure 10-28.

VIEWS	Top	Docs	Category	Time Series	Topic	Delta	2D Map
<input type="checkbox"/>	VISA processing			17			
<input type="checkbox"/>	VISA requirement			15			
<input type="checkbox"/>	visa			15			
<input type="checkbox"/>	VISA APPLICATION			15			
<input type="checkbox"/>	visa application			15			
<input type="checkbox"/>	visa holder			15			
<input type="checkbox"/>	embassy			15			
<input type="checkbox"/>	VISA application			15			
<input type="checkbox"/>	stay			15			

Figure 10-28 Having both a rule and Dictionary entries gets better results

10.4.4 Sample rules

There are many rules we can apply to analyze e-mail data. Although we do not have data to cover all examples, we include sample rules that you can use to get additional information.

Sample rules: Capture question related information

You can create rules to capture questions people have associated with a particular business process, project, or practice.

If a person says “I need information on <something>”, then we know that person has a question on a particular procedure, subject matter, or things. We can create a rule to capture this expression as follows:

```
<mi category=".visa.requirements.question.phrase" name="" value="My
Question is ${3.str} ${4.str} ${5.str} ${6.str} ${7.str}">
<w id="0" lex="I"/>
<w id="1" lex="need"/>
<w id="2" lex="information"/>
<w id="3"/>
<w id="4"/>
<w id="5"/>
<w id="6"/>
<w id="7"/>
</mi>
```

Notice that the token #3, 4, 5, 6, and 7 can be anything the user enters. When displaying the result set in Text Miner, the values for these tokens are displayed in the view for analysis.

People can ask questions in many ways other than what we describe above. When they have questions, they might say: “I got a question <on, about, regard, of something>”, “I have a question <on something or of something>”, or “I have several questions <on something or of something>.” To capture this type of expression, we can create a rule as follows:

```
<mi category=".visa.requirements.question.phrase" name="" value="My
Question is ${3.str} ${4.str} ${5.str} ${6.str} ${7.str}" ">
<w id="0" lex="I"/>
<w id="1" lex="/(^have$)|(^got$)/"/>
<w id="2"/>
<w id="3" lex="question"/>
<w id="4"/>
<w id="5"/>
<w id="6"/>
<w id="7"/>
</mi>
```

Notice that the token #2 in a sentence can be anything. This is a place holder in case the person writes “I have *a* question” or I have *one* question”. Because token #3 has a lex “question”, the pattern also captures “I have *several questions*”, “I have *few questions*”, or “I have *many questions*”. Token #4, 5, 6, and 7 can also be anything user enters.

Another generic expression that people use when they have questions is in the form of “additional information needed <for something or of something>.” To capture this type of expression, we can create the following rule:

```
<mi category=".visa.requirements.question.phrase" name="" value="My
Question is ${3.str} ${4.str} ${5.str} ${6.str} ${7.str}" ">
<w id="0" lex="additional"/>
<w id="1" lex="information"/>
<w id="2" lex="need"/>
<w id="3"/>
<w id="4"/>
<w id="5"/>
<w id="6"/>
<w id="7"/>
</mi>
```


Sample rules: Capture people who are unable to do something

By understanding what people cannot do, it will give us insights as what the problems might be that are associated with the Visa application process or other business process.

To capture a common phrase such as “<someone> can not <do or perform> <something>”, for example, “I can not submit the form” or “I can not handle the process”, we can use the following rule:

```
<mi category=".visa.requirements.question.phrase" name="" value="unable
to ${2.lex} ${n1.lex}">
<w id="0" lex="can"/>
<w id="1" lex="not"/>
<w id="2" lex="!be"/>
<w id="3"/>
<w id="n1" pos="noun"/>
</mi>
```

Note that if a sentence contains “cannot” or “could not”, it will break down to “can not”. Thus, this rule would capture sentences such as “I cannot submit the form” or “I could not handle the process.”

Also note that the token #3 from the rule can be anything, for example, “a” or “the”. This token is not important in categorizing this common phrase. Therefore, it is not part of the rule value.

The last token has to be a noun. If the user enters words other than a noun, then the rule would not capture it. To capture sentences such as “<someone> can not <do whatever other than a something>”, we can add a generic rule in addition to the previous one:

```
<mi category=".visa.requirements.question.phrase" name="" value="unable
to ${2.lex}">
<w id="0" lex="can"/>
<w id="1" lex="not"/>
<w id="2" lex="!be"/>
</mi>
```

Notice that the first and second rules can apply to the same sentence. Practical rules need guard words to suppress such duplicate captures. Pay attention to how you order the rule pattern in the rule file. The system evaluates rules according to the order you put in.

Sample rules: Capture negative phrases

To capture mood of people in e-mails, you might want to create rules associated with negative phrases.

A general phrase people use when they are not happy or satisfy with something is "... not happy with <something>" or "... not satisfied with <something>". To capture this type of expression, we can use the following rule:

```
<mi category=".visa.requirements.negative.phrase" name="" value="not
${1.str} with ${n1.lex}">
<w id="0" lex="/(not$)|(isnt$)|(arent$)"/>
<w id="1" lex="/^((happy)|(satisfy)|(satisfactory))$/"/>
<w id="2" lex="with"/>
<w id="n1" pos="noun"/>
</mi>
```

Note that if a sentence contains "isn't" and "aren't", it will be broken down to "is not" and "are not" and would be captured from the (not\$) rule. The "isnt\$" and "arent\$" in the rule captures "isnt" and "arent" denoted literally in a sentence. So, for example, the above rule not only can capture a sentence such as "this is not satisfactory", "this isn't satisfactory", the rule can also capture a sentence such as "this isnt satisfactory".

Also, the (satisfy) in the rule captures sentences with the word "satisfy" or "satisfied". Thus, "I am not satisfied" would be captured by this rule as well.

Also notice that the last token is any noun. Sometimes, people might put something before a noun. For example, people might say "I'm not happy with *the* application" or "I'm not satisfied with *the* process". We can add additional rule to capture this type of express:

```
<mi category=".visa.requirements.negative.phrase" name="" value="not
${1.str} with ${n1.lex}">
<w id="0" lex="/(not$)|(isnt$)|(arent$)"/>
<w id="1" lex="/^((happy)|(satisfy)|(satisfactory))$/"/>
<w id="2" lex="with"/>
<w id="a1" pos="/^((determiner)|(adjective))$/"/>
<w id="n1" pos="noun"/>
</mi>
```

What if people did not say "... not happy *with* <something>" or "... not satisfied *with* <something>"? In other words, people might not say "with" in the sentence. They might say "of <something>". We can use a generic rule to capture anything that they say that resemble "... not happy <anything>" or "... not satisfied <anything>". We can use the following rule to capture this generic expressions in addition to previous ones:

```
<mi category=".visa.requirements.negative.phrase" name="" value="not
${1.lex}">
<w id="0" lex="/(not$)|(isnt$)|(arent$)"/>
<w id="1" lex="/^((happy)|(satisfy)|(satisfactory))$/"/>
</mi>
```

Sample rule: Capture generic trouble words

You might want to create a list of generic trouble words and associate it with other keywords for high frequencies and high correlation analysis in your e-mail data or any type of data that contain people's opinion or description of process, events, or things in general. The following is a sample rule that captures generic bad words, such as "anger", "angry", "complain", "condemnd" (for possible wrong spelling), "condemned", "condemn" (for other possible wordings), "insult", "yelling", "upset", and more:

```
<mi category=".visa.requirements.negative.trouble.word" name=""
value="{1.lex}">
<w id="1"
lex="/^((anger)|(angry)|(aplgzd)|(apologize)|(apologized)|(complain)|(c
omplained)|(condemd)|(condemn)|(condemned)|(disagree)|(disgruntle)|(dis
gruntled)|(disturb)|(error)|(frustrate)|(frustrated)|(harrasmnt)|(harra
ssing)|(harrassment)|(inconvenience)|(insult)|(irate)|(mad)|(maddening)
|(offend)|(sorry)|(threat)|(threatened)|(tired)|(trouble)|(unhappy)|(un
satisfactory)|(unsatisfied)|(upset)|(upst)|(yell)|(aggravated)|(aggreva
ted)|(aggravated)|(aggrvtd)|(disturbing))$/"/>
</mi>
```

When creating rules to capture moods or problems people encounter, also remember that we are dealing with people here. People are diverse in the way they use language. As analysts, we need to be descriptive, not prescriptive.

10.5 Apply new rules and dictionaries

Whenever new rules or dictionaries are created by using the TAKMI DICTIONARY application, a new OmniFind Analytics Edition index has to be built to reflect the new associations of keywords, rule, and categories.

The following is the required steps you should perform after creating a new rule or a new dictionary category or keywords:

1. Stop WebSphere Application Server.

Once the WebSphere Application Server is stopped, the TAKMI_MINER_EAR (Text Miner) application that holds the handles to the OmniFind Analytics Edition index files are released and the index file can be rewritten. If your WebSphere Application Server is running other critical applications and you do not want to interrupt their runtimes, then you can launch the administration console for the WebSphere Application Server and manually stop TAKMI_MINER_EAR application.

In Figure 10-29, we show the WebSphere administration console with the stopped state of TAKMI_MINER_EAR application.

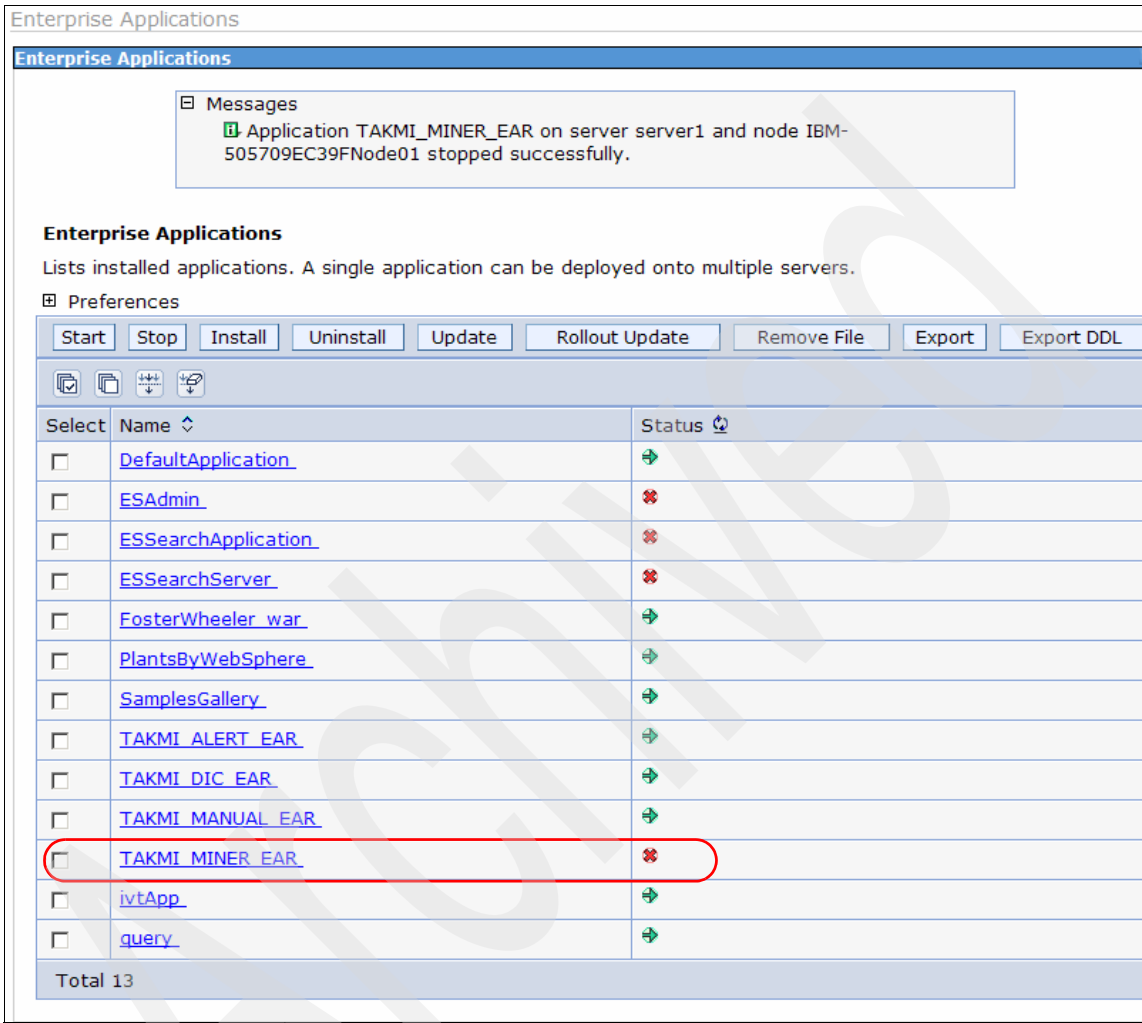


Figure 10-29 Stopped status of TAKMI_MINER_EAR (Text Miner) application

2. Preserve or recreate ATML files.

If the source data has not been changed, you will not have to recreate the ATML files from the source CSV files.

A recreation of the ATML files is required only if new data or updates have been committed to the source data. Refer to 4.5.2, “Invoke Data Ingestor command to convert CSV to ATML format” on page 77 to create new ATML files.

3. Delete index and MIML files.

Refer to 4.9, “Deleting data” on page 84 to learn the steps to delete the existing MIML and index files.

4. Recreate new MIML and index files.

To recreate the MIML and index files, refer to 4.6, “Natural language processing” on page 78 and 4.7, “Indexing” on page 81.

The creation of the new MIML files involves the language processing annotators and application of the new dictionaries and rules to the data that will eventually get indexed.

5. Once the index is built, restart the TAKMI_MINER_EAR (Text Miner) application.

You can restart it from the administration console of WebSphere Application Server or simply restart the WebSphere Application Server instance.

In Figure 10-30, we show the administration console of the WebSphere Application Server with the running status of TAKMI_MINER_EAR application.

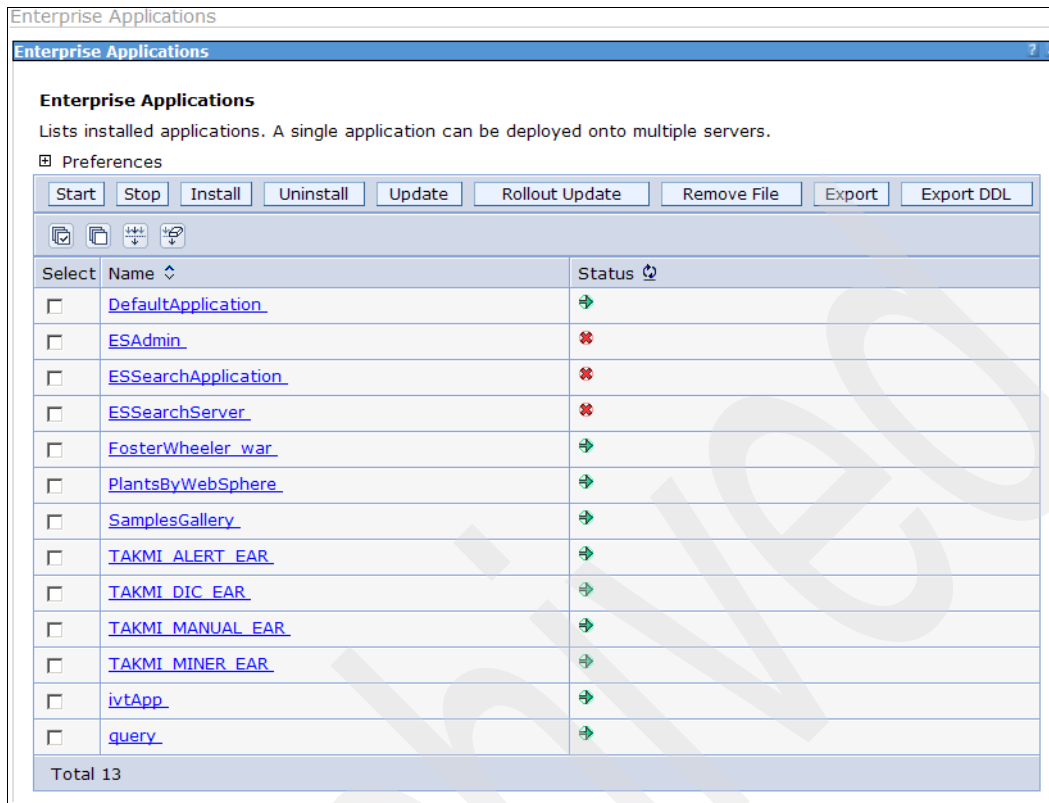


Figure 10-30 Start status of TAKMI_MINER_EAR (Text Miner) application

6. Access the TAKMI_MINER_EAR (Text Miner) application to validate your changes.

Integrating OmniFind Analytics Edition with OmniFind Enterprise Edition

By default, IBM OmniFind Analytics Edition (OAE) uses CSV formatted files as input files for content processing and analytics. OAE does not use crawlers to pull enterprise content. To leverage the wide variety of crawlers provided by IBM OmniFind Enterprise Edition (OEE), OAE distributes a UIMA pear annotator, to be used by OEE, to convert the crawled data to OAE required data format. In this chapter, we elaborate the integration of IBM OmniFind Enterprise Edition and IBM OmniFind Analytics Edition. We discuss the creation of an index for documents crawled by OEE.

We cover the following topics:

- ▶ Introduction to IBM OmniFind Enterprise Edition
- ▶ Integration of OAE and OEE
- ▶ Uploading OmniFind Analytics Edition annotator
- ▶ Building analytics index
- ▶ Registering and browsing analytics index
- ▶ Integrating custom dictionaries

11.1 Introduction to IBM OmniFind Enterprise Edition

IBM OmniFind Enterprise Edition (OEE) delivers full-scale secure enterprise search capabilities. Built on a flexible, open architecture, it offers a powerful combination of performance, security, scalability, enterprise reach, and openness for applying advanced linguistic processing. By simply entering a keyword or phrase in a query, users can quickly search across intranets, corporate public Web sites, relational database systems, file systems, and content repositories, and obtain a consolidated, ranked meaningful result. Users also have the ability to go beyond standard full-text search and perform parametric and semantic queries to dramatically improve the relevancy of search results.

Predefined support is available for searching a variety of data source types. OmniFind Enterprise Edition can collect data from the following types of data sources:

- ▶ IBM Content Manager item types (documents, resources, and items)
- ▶ IBM DB2 Universal Database™ (DB2 UDB) databases
- ▶ IBM Lotus Notes databases
- ▶ IBM Lotus QuickPlace® databases
- ▶ IBM WebSphere Information Integrator Content Edition repositories: Documentum, IBM FileNet Panagon Content Services, IBM FileNet Content Manager, Hummingbird Document Management (DM), Microsoft SharePoint®, OpenText Livelink, and Portal Document Manager (PDM) item classes
- ▶ IBM WebSphere Information Integrator nickname tables for many database system types, including IBM DB2 UDB for z/OS®, IBM Informix, Microsoft SQL Server®, Oracle®, and Sybase
- ▶ IBM WebSphere Portal sites
- ▶ IBM Workplace™ Web Content Management sites
- ▶ Microsoft Exchange Server public folders
- ▶ Microsoft SQL Server databases
- ▶ Microsoft Windows file systems
- ▶ Network news transfer protocol (NNTP) news groups
- ▶ Oracle databases
- ▶ UNIX® file systems
- ▶ Web sites on the Internet or intranet

You can also add support for searching the following types of external sources:

- ▶ Databases that support the Java database connectivity (JDBC™) protocol (DB2 UDB and Oracle database systems only). A separate external source is created for each table that you enable for searching.
- ▶ Lightweight Directory Access Protocol (LDAP) servers. One external source is created for each LDAP server.

11.2 Integration of OAE and OEE

In IBM OmniFind Analytics Edition, the data to be analyzed, including text, should be prepared in Comma Separated Values (CSV) data format. Most of the relational databases and spreadsheet applications support exporting data to CSV format. But if the data to be extracted is not in a spreadsheet, then the extraction of data to the CSV format might become tedious and complicated.

In these situations, where most of the data to be analyzed is lying in different content repositories and scattered across the enterprise such as IBM Content Manager, Document Manager, emails, Web sites, and portals, OmniFind Analytics Edition can be integrated with OmniFind Enterprise Edition to handle the data extraction part from the different repositories. OmniFind Enterprise Edition extracts the data from different repositories. The natural language processor (NLP) UIMA annotators parse the extracted data and convert them to MIML files. The natural language processor (NLP) UIMA annotators are distributed with the OmniFind Analytics Edition installation.

This integration leverages IBM OmniFind Enterprise Edition's capabilities to crawl a wide range of data sources and support for integrating custom UIMA annotators. As an example, in this chapter, we use OEE's Lotus notes crawler to crawl a sample notes email database and the OmniFind Analytics Edition's natural language annotators to generate a OmniFind Analytics Edition index.

Figure 11-1 illustrates a high level view of the integration mechanism between OmniFind Analytics Edition and OmniFind Enterprise Edition.

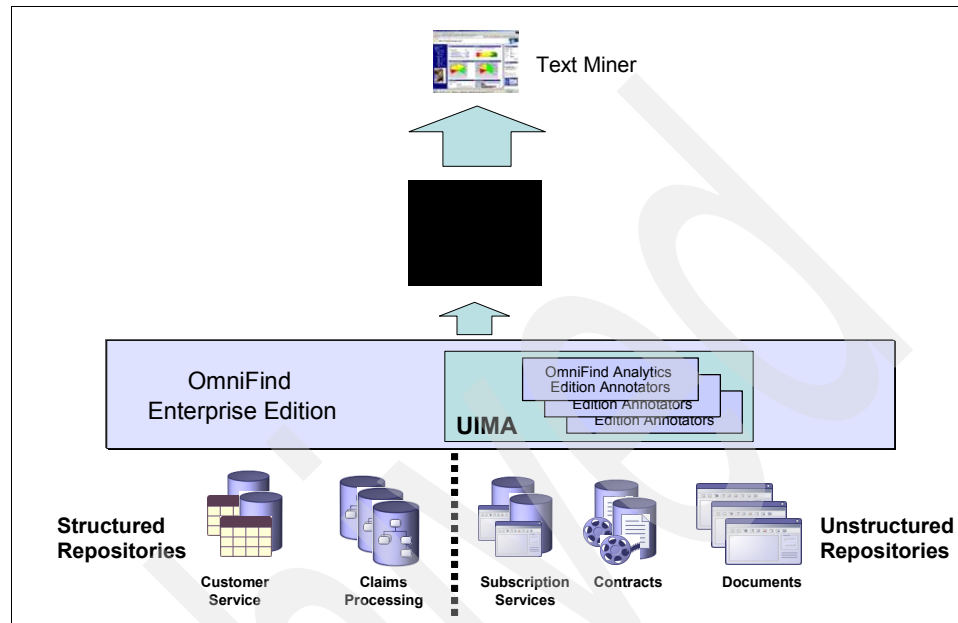


Figure 11-1 Overview architecture of OAE and OEE integration

The integration is done by registering UIMA annotators for natural language processing into OmniFind Enterprise edition. These annotators parse the data extracted by crawlers to generate MIML files. The MIML files contain the source content, the metadata discovered by the OEE crawlers, and the metadata discovered by OmniFind Analytics Edition's language processing annotators. The generated MIML files will then be used to generate the OmniFind Analytics Edition index. Figure 11-2 shows the detailed flow of data across various components of OEE and OmniFind Analytics Edition.

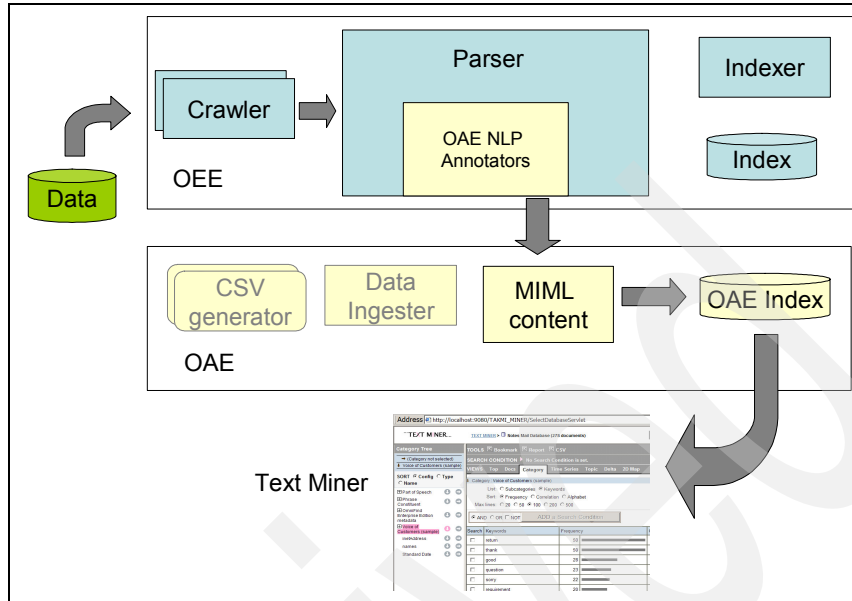


Figure 11-2 Detailed data flow

In the following sections, we describe the steps required to integrate OmniFind Analytics Edition with OmniFind Enterprise Edition:

1. Uploading OmniFind Analytics Edition annotator.
2. Building analytics index.
3. Registering and browsing analytics index.
4. Integrating custom dictionaries (optional).

11.3 Uploading OmniFind Analytics Edition annotator

OmniFind Analytics Edition provides UIMA based natural language processing annotators to support English and Japanese languages. They can be found at the following location:

- ▶ <TAKMI_HOME>\pear\NLP_PACK_en.pear
- ▶ <TAKMI_HOME>\pear\NLP_PACK_ja.pear

Note: <TAKMI_HOME> is the install directory for OmniFind Analytics Edition product.

Before you upload the provided pear file into OmniFind Enterprise Edition, it is mandatory to meet the following prerequisites:

1. Launch the administration console for OEE.
2. Identify or create a new collection in OEE.
3. Create or identify a crawler instance that will extract your source repository. This could be a Web site or Notes Email database or file system or FileNet CM or any other repositories supported by OEE.

11.3.1 Uploading and associating UIMA annotators

The following steps elaborate the instructions required to upload the UIMA annotators and associate it with a collection.


1. Copy or FTP NLP_PACK_en.pear to a temporary directory that is local to the server where IBM OmniFind Enterprise Edition is installed. We created a directory called <OEE_INSTALL_ROOT>\pear directory and copied NLP_PACK_en.pear.

Note:








<OEE_INSTALL_ROOT> is the install directory for OmniFind Enterprise Edition.

Since OmniFind Enterprise Edition does not support dynamic uploads of pear files larger than 8 MB, it is mandatory to provide the pear file local to OEE installation.

2. Launch the administration console for IBM OmniFind Enterprise Edition.
3. Access the configuration page where the pear files can be registered. The configuration page can be accessed by clicking **System tab** → **Edit Mode** → **Parse tab** → **Configure test analysis engines**. Figure 11-3 shows the related page.

Address  http://localhost:9081/ESAdmin/system.do?command=addPear

OmniFind Enterprise Edition

 Collections
  External Sources
  System
  Security
  Search Customizer
  Log Out
  ?

[System : Parse](#) > [Configure text analysis engines](#) > [Add Text Analysis Engine](#)

Add a Text Analysis Engine

[Help for this page](#) ?

Type a unique display name for the text analysis engine that you want to add to the system. When you configure text processing options for a collection, you select the engine that you want to use. Specify the file path:

- If the processing engine archive (.pear file) is on your local system, you can click Browse to
- If the .pear file is on the index server, type the fully qualified path for the file.

Text analysis engine name:

☐ Local path (the maximum file size is 8 MB)

File name:

☒ Index server path (the file can be larger than 8 MB)

File name:

Figure 11-3 Register PEAR file with OEE

- As displayed in Figure 11-3, confirm a name for the text analysis engine and the absolute path of the pear file (NLP_PACK_en.pear). At the end of the registration, confirm pear contents are installed at <OEE_INSTALL_ROOT>/data/pearsupport/PearIDN, where **N** is a unique number to identify pear modules in the index server. In the rest of the section, we will refer to this directory as <PEAR_ROOT>
- If OmniFind Analytics Edition installation is not installed on the OmniFind Enterprise Edition server, update the system library path to contain <PEAR_ROOT>/bin.

Note: System library path for various operation systems:

- ▶ For windows, the environment variable for system library path is %PATH%.
- ▶ For AIX, the environment variable for system library path is \$LIBPATH.
- ▶ For Linux®, the environment variable for system library path is \$LD_LIBRARY_PATH.

6. Now that OEE has successfully registered the pear file, we can now assign the analysis engine to the collection identified earlier in this section. Access the parser configuration page for the identified collection. The configuration page can be accessed by clicking **Collection tab** → **Parse button** (for the identified collection) → **Edit mode** → **Configure text processing options**. Click on **Select a text analysis engine** to access the collection and PEAR association page. From the drop down menu select the analysis engine that was uploaded in step 3 on page 300. Figure 11-4 shows the Text Processing Options page.

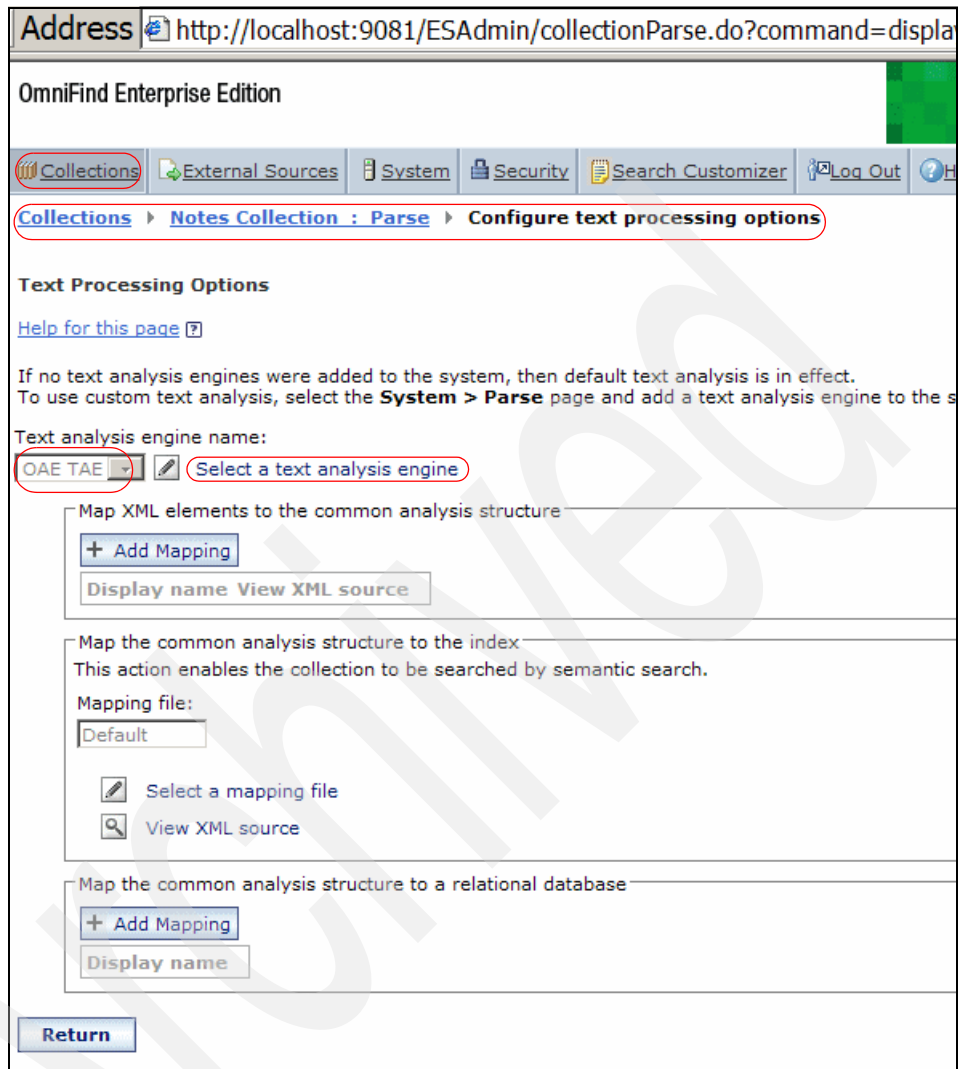


Figure 11-4 Text Processing Options page

11.3.2 Start crawlers and parse

With the annotator in place and the crawler instance defined to access your target data, you are ready to crawl and parse the documents. At the end of parsing you should find the MIML files generated at <PEAR_ROOT>\database\db\miml.

To crawl and parse the documents:

1. Access the monitor pages for the crawler and invoke a *full crawl* of the discovered target data. Monitor the crawler statistics to determine the completion of the crawl.

Refer to the following URL for monitoring crawlers:

<http://publib.boulder.ibm.com/infocenter/discover/v8r4/index.jsp?topic=/com.ibm.discovery.es.ad.doc/monitoring/iisatmcrawl.htm>

2. When the crawl is complete, start the collection parser to allow the associated analysis engine to create MIML files.

Refer to the following URL for monitoring parsers:

<http://publib.boulder.ibm.com/infocenter/discover/v8r4/index.jsp?topic=/com.ibm.discovery.es.ad.doc/monitoring/iisatmparse.htm>

3. When the parsing is complete, confirm the MIML files are generated at <PEAR_ROOT>\database\db\miml.
4. Close the administration console for OmniFind Enterprise Edition. We do not need to execute the indexing operation on OmniFind Enterprise Edition. From this point, we no longer have to access OmniFind Enterprise Edition.

11.4 Building analytics index

With the MIML files generated by the OEE parser, we can now invoke the scripts to generate an index compatible for OmniFind Analytics Edition.

Example 11-1 shows the output of the index build command.

Example 11-1 Output for takmi_index.bat script

```
C:\>takmi_index.bat
C:\Progra~1\IBM\es\test\data\pearsupport\PearId1\database
database_config.xml has been generated successfully.
***** ABOUT LOG FILE *****
*****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern
* property of
*   C:\Program Files\IBM\takmi\conf\TAKMI_indexer_logging.properties
* file.
* The default location is C:\Program
Files\IBM\takmi\logs\TAKMI_indexer_batch.log .
*****
*
```



```

DataEntry[db\miml\docset_20071010_131037_0.miml] processing start.
***
DataEntry[db\miml\docset_20071010_131037_0.miml] processing done.
***** ABOUT LOG FILE *****
*****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern
* property of
*   C:\Program Files\IBM\takmi\conf\TAKMI_indexer_logging.properties
* file.
* The default location is C:\Program
Files\IBM\takmi\logs\TAKMI_indexer_batch.log .
*****
*
File Merge start.
***** ABOUT LOG FILE *****
*****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern
* property of
*   C:\Program Files\IBM\takmi\conf\TAKMI_indexer_logging.properties
* file.
* The default location is C:\Program
Files\IBM\takmi\logs\TAKMI_indexer_batch.log .
*****
*
File Merge done.
Group Merge start.
.....
***** ABOUT LOG FILE *****
*****
* The location of the log file is specified by
*   java.util.logging.FileHandler.pattern
* property of
*   C:\Program Files\IBM\takmi\conf\TAKMI_indexer_logging.properties
* file.
* The default location is C:\Program
Files\IBM\takmi\logs\TAKMI_indexer_batch.log .
*****
*
Group Merge done.
Index Build done.
C:\>

```

Note: For the takmi_index command, specify <PEER_ROOT>\database as the argument.

11.5 Registering and browsing analytics index

When the index is built, refer to 4.2, “Database directory creation and registration” on page 58 and 4.8, “Accessing Text Miner application” on page 83 to register and browse the index.

In the global_config.xml, specify absolute path for the database. Refer to the highlighted text in Example 11-2.

Example 11-2 global_config.xml with the new index

```
<?xml version="1.0" encoding="UTF-8"?>
<global_config>

<params>
<param name="language" value="en"/>
</params>

<database_entries>
<database_entry name="INDEXED_DATA_SAMPLE_EN" path_type="relative"
path="databases/INDEXED_DATA_SAMPLE_EN"/>
<b><database_entry name="Notes Mail Database" path_type="absolute"
path="C:/Progra~1/IBM/es/test/data/pearsupport/PearId1/database" />
</database_entries>
</global_config>
```

Figure 11-5 and Figure 11-6 illustrates the successful registration and navigation of the newly created index.

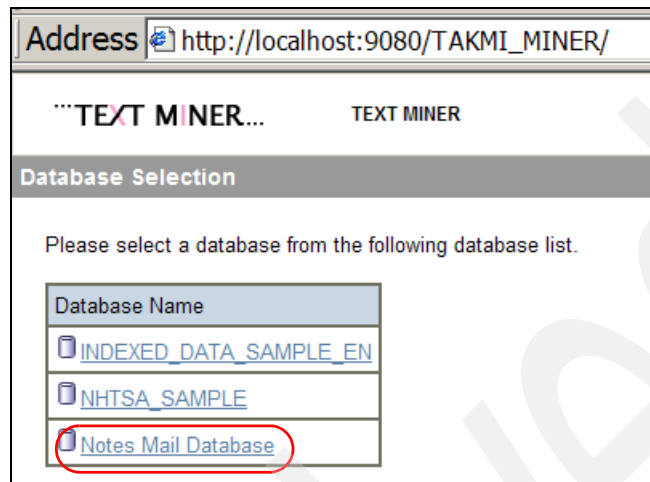


Figure 11-5 Text miner window with the new index

Click **Notes Mail Database** in Figure 11-5 to access the Category view as shown in Figure 11-6. The Figure 11-6 we have highlighted a category called OmniFind Enterprise Edition metadata that represents the metadata fields extracted from the OEE crawlers. Figure 11-6 also shows the system categories (Part of speech, phrase Constituent) discovered by the OmniFind Analytics Edition natural language annotators.

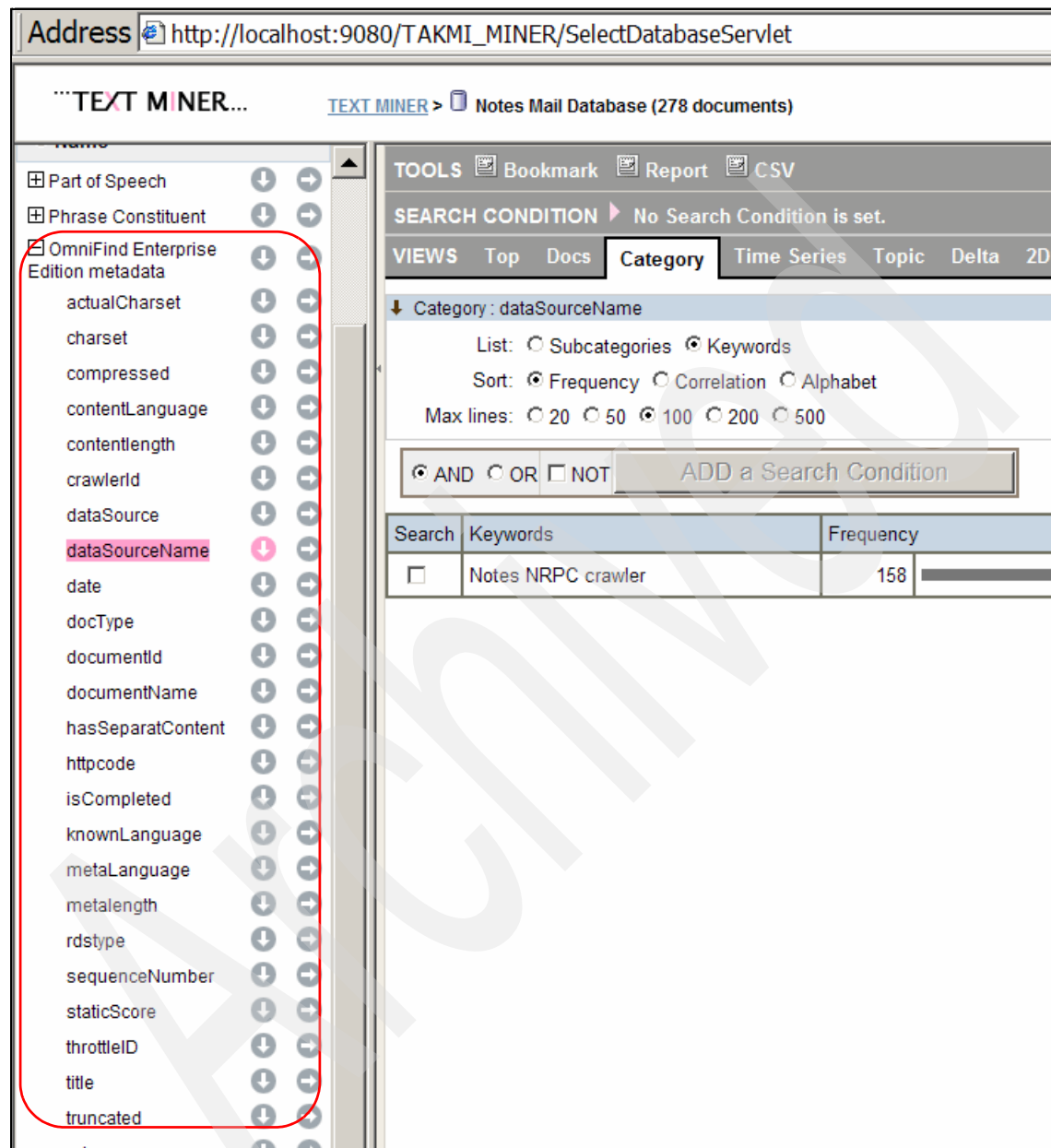


Figure 11-6 Text miner window with indexed data

11.6 Integrating custom dictionaries

The categories displayed in Figure 11-6 on page 308 are the direct results of applying the default language dictionaries to the OmniFind Analytics Edition index. In this section, we explain the integration of custom dictionaries such that user defined categories and associated keywords are mapped for text analytics.

11.6.1 Creating a custom dictionary

After analyzing the index created from a Notes mail database, we discovered a need to identify documents by the physical user names participating in a mail.

The sample mail database consists of email transactions related to group of engineers across the world who have to visit Japan for 5 weeks for a Redbooks publication residency program. For non-US citizens, a short term Visa is required to visit Japan. With the VISA restrictions in view, we also felt a need to identify emails that referred to immigration.

To accomplish the above requirements, we use Dictionary Editor to create two categories, namely immigration and names in OmniFind Analytics Edition. In Figure 11-7, we display the two category names.

Category Tree	
Add Root Category	
Immigration	Add Subcategory Rename Delete
names	Add Subcategory Rename Delete

Figure 11-7 Immigration and names categories

We assigned a list of keywords and synonyms that was relevant with the terms used in embassies. As shown in Figure 11-8, we display a sample set of keywords/synonyms and their association to category called Immigration.

Display Mode
Category Tree

RESET CATEGORY SEARCH

Category Tree

Immigration
ADD

Registered Keyword List (9 keywords)

1 - 9
SELECT ALL
NEW KEYWORD

Select	POS	Keyword	Synonym	Ca
<input type="checkbox"/>	Noun	air ticket	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	citizen	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	travel agent	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	business	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	invitation letter	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	immigration	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	foreigner	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	VISA =visa	EDIT	Immigration REMOVE
<input type="checkbox"/>	Noun	embassy	EDIT	Immigration REMOVE

Figure 11-8 Keywords/Synonyms for Immigration category

For the names category, we assigned names of users who participated in the residency program. In Figure 11-9, we show the sample list of user names marked as keywords and their association to the names category.

Display Mode
Category Tree

RESET CATEGORY SEARCH

Category Tree

Immigration
ADD

names
ADD

Registered Keyword List (8 keywords)

1 - 8
SELECT ALL
NEW KEYWORD

Select	POS	Keyword	Synonym	Ca
<input type="checkbox"/>	Noun	Shin Takakura	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Iwao Inagaki	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Kameron Cole	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Ratheesh Muralaedharan	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Scott Harms	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Srinivas Chitivel =varma	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Deanna Polm	EDIT	names REMOVE
<input type="checkbox"/>	Noun	Wei-Dong Zhu =Jackie	EDIT	names REMOVE

Figure 11-9 participant names registered as keywords for the names category

When the new additions are saved, we need to invoke the TAKMI batch program to update the default LanguageWare dictionary with the new additions. The default LanguageWare dictionary is located at DATABASE_DIR\dic\LangWare50\en-XX-TAKMIUserNE.dic.

Example 11-3 shows the sample output of the TAKMI resource deploy command that updates en-XX-TAKMIUserNE.dic.

Example 11-3 Sample output for takmi_nlp_resource_deploy.bat

```
C:\Program
Files\IBM\es\test\data\pearsupport\PearId1>takmi_nlp_resource_deploy
.bat "C:\Program Files\IBM\es\test\data\pearsupport\PearId1\database"
word boundary dictionary conversion
AnalyzerDictionaryConversion for C:\Program
Files\IBM\es\test\data\pearsupport\
PearId1\database
Conversion completed
dictionary pattern generation
AnalyzerDictionaryPatternGenerator for C:\Program
Files\IBM\es\test\data\pearsu
pport\PearId1\database
Pattern generation completed
English mode [LW binary dictionary generation]
IBM LanguageWare XML Dictionary Builder build 20050831.417
Loading schema:
file:/C:/Program%20Files/IBM/es/test/data/pearsupport/PearId1/d
atabase/dic/LangWare50/named_entity.xsd
Preparing XML parser
Parsing dictionary source: C:\Program
Files\IBM\es\test\data\pearsupport\PearId
1\database\dic\LangWare50\takmiUserDic.xml
Processed 40 words
Saving dictionary C:\Program
Files\IBM\es\test\data\pearsupport\PearId1\datas
e\dic\LangWare50\en-XX-TAKMIUserNE.dic
Finished
C:\Program Files\IBM\es\test\data\pearsupport\PearId1>
```

11.6.2 Registering OAE LanguageWare dictionary with OEE

The parsers in OEE uses a language dictionary to parse the source content into metadata fields. OEE provides external configuration files to provide additional LanguageWare dictionaries.

To register the updated OmniFind Analytics Edition dictionary with OEE so that the keywords in the OmniFind Analytics Edition dictionary are used in addition to the OEE dictionaries:

1. Stop the OEE server. Refer to the OEE administration guide to stop all the processes that relate to a working OEE server.
2. Copy <PEAR_ROOT>/database/dic/LangWare50/en-XX-TAKMIUserNE.dic to <OEE_INSTALL_ROOT>/configurations/parserservice/jediidata/frost/resources directory.
3. In order to recover from manual mistakes, we recommend to take a backup of the file called \$ES_NODE_ROOT/master_config/collection_id.parserdriver/specifiers/jfrost.xml where collection_id is the collection ID of the target collection. In our scenario, we assume the collection ID of the target collection is notes.
4. Update \$ES_NODE_ROOT/master_config/collection_id.parserdriver/specifiers/jfrost.xml with en-XX-TAKMIUserNE.dic.

Example 11-4 shows the sample contents of jfrost.xml with changes highlighted in bold.

Example 11-4 Updated contents of jfrost.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<taeDescription xmlns="http://uima.watson.ibm.com/resourceSpecifier">
..
..
  <!-- English -->
  <settingsForGroup name="en">
    <nameValuePair>
      <name>LexicalDicts</name>
      <value>
        <array>
          <!-- dictionary additions for the OAE index -->
          <string>en-XX-TAKMIUserNE.dic</string>
          <string>en-XX-Lex.dic</string>
        </array>
      </value>
    </nameValuePair>
    <nameValuePair>
      <name>StopwordDicts</name>
      <value>
        <array>
          <string>en-Stw.dic</string>
```



```

        </array>
      </value>
    </nameValuePair>
    <nameValuePair>
      <name>SpellCorrectionDicts</name>
      <value>
        <array>
          <string>en-XX-Rules.dic</string>
        </array>
      </value>
    </nameValuePair>
  </settingsForGroup>
..
..
</taeDescription>

```

5. While working with OEE, we discovered that the parser breaks URLs and mail addresses into word pieces; For example “somebody@jp.ibm.com” will be the word sequence of “somebody”, “@”, “jp”, “.”, “ibm”, “.”, and “com”. this made it impossible to search documents by Internet email addresses.

To prevent the URL/EMAIL segmentation by OEE parser, set the value of DoURISegmentation element to false. Example 11-5 shows the changes highlighted in bold.

Example 11-5 Prevent URL segmentation

```

<?xml version="1.0" encoding="UTF-8"?>
<taeDescription xmlns="http://uima.watson.ibm.com/resourceSpecifier">
..
..
<nameValuePair>
  <name>DoURISegmentation</name>
  <value>
    <!-- Prevent segmentation of URLs and mail addresses into words -->
    <boolean>false</boolean>
  </value>
</nameValuePair>
..
..
</taeDescription>

```

The only drawback by the change is, the query “somebody” will not match “somebody@jp.ibm.com”.

6. Because we do not have new data but we are only trying to apply new dictionaries, duplicate data can be eliminated by deleting the existing MIML and index files. In Example 11-6, we show the sample output of the delete command.

Example 11-6 Sample output of takmi_clear_nlp_index.bat

```
C:\Program
Files\IBM\es\test\data\pearsupport\PearId1>takmi_clear_nlp_index.bat
"C:\Program Files\IBM\es\test\data\pearsupport\PearId1\database"
***** WARNING *****
* THIS COMMAND CLEARS ALL MIML FILES *
* AND INDEX FILES.                  *
*****
OK? Please enter Y or N :y
directory "C:\Program
Files\IBM\es\test\data\pearsupport\PearId1\database\db\in
dex" and all miml files in "C:\Program
Files\IBM\es\test\data\pearsupport\PearI
d1\database\db\miml" were successfully deleted.
C:\Program Files\IBM\es\test\data\pearsupport\PearId1>
```

7. Start the OEE server. After registering the updated OmniFind Analytics Edition dictionary, we will have to recrawl and reparse the mail database. These actions will trigger the application of new keywords and categories to the new set of MIML files generated by the OEE parse. Refer to the section on “Start crawlers and parse” for re crawling and reparsing.

11.6.3 Rebuilding the OmniFind Analytics Edition index

With the new set of MIML files we can now build a new OmniFind Analytics Edition index. Refer to 11.4, “Building analytics index” on page 304 where we explain the creation of OmniFind Analytics Edition index from the MIML files.

After a new index is created, access the Text Miner application to confirm the discovery of new categories and keywords.

In Figure 11-10, we see the new categories called immigration and names.

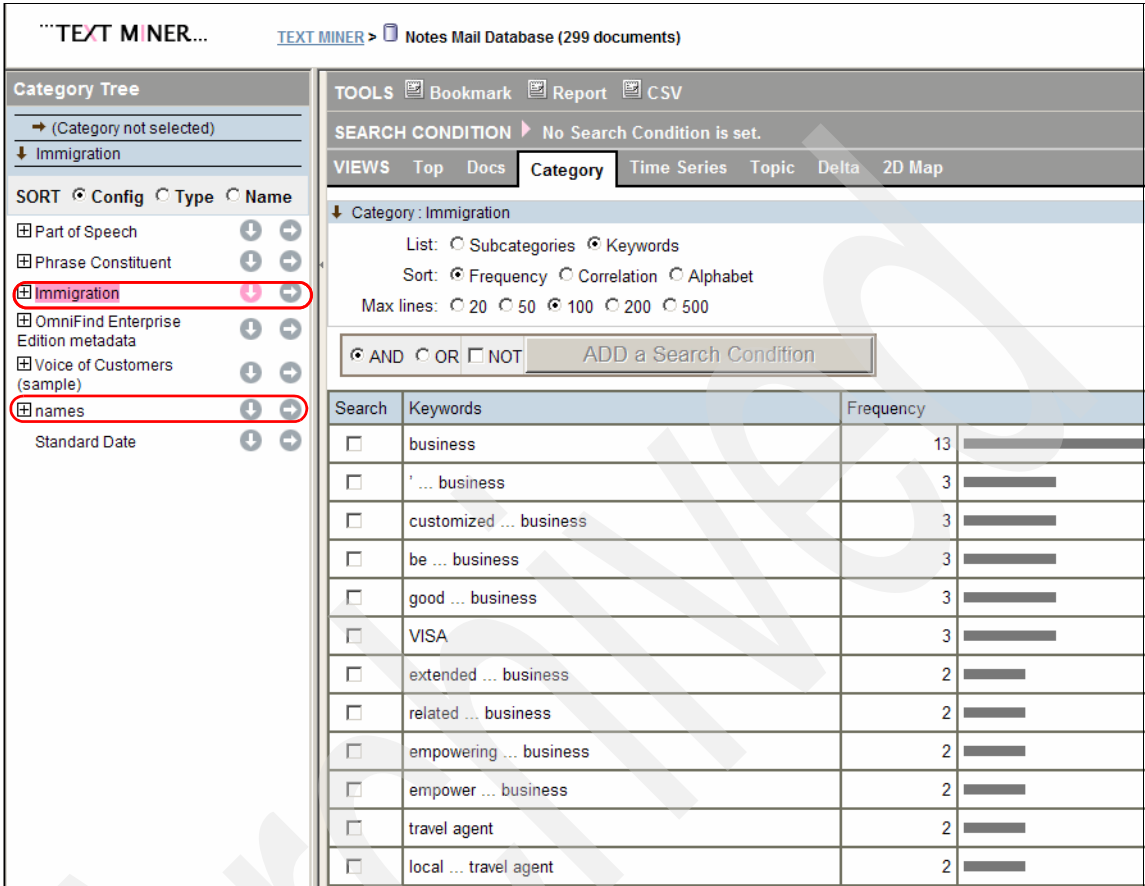


Figure 11-10 New categories in Text miner application

With the new categories as Figure 11-10, we can easily identify documents that refer to immigration issues and also possible to identify the users involved in any of the emails.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

Online resources

These Web sites are also relevant as further information sources:

- ▶ OmniFind Analytics Edition support page:
<http://www.ibm.com/software/data/enterprise-search/omnifind-analytics/support.html>
- ▶ Open source platform Unstructured Information Management Architecture (UIMA), developed by IBM Research:
<http://incubator.apache.org/uima>

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications, and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

- 2D Heat Map view 18, 23
- 2D Map view 10, 94–97, 109, 111, 128, 130, 148, 230, 243, 272
 - correlation numbers 131
 - multiple categories 129
 - understanding 131

A

- Advanced Text analytic Markup Language (ATML) 15, 57
- air condition (AC) 176
- airbag 176
- airbag deployment 7, 230
- AIX environment 32, 46
- Alerting System 25, 50, 52, 127, 131, 222
 - batch processing 235
 - setting up 222
- analysis
 - delta 9–10
 - fire incident, case study 271
 - statistical and linguistic 8
 - trend 9–10
- analysis result 18, 24, 134, 235–236, 262
- analytic application 17
- analyze data versus mine data 4
- annotation 192
- annotator 18, 214
 - uploading from OmniFind Analytics Edition 299
- application verification 50–51
- architecture
 - OmniFind Analytics Edition and OmniFind Enterprise Edition integration 298
- artisan 2
- association
 - keyword to category 64
- ATML file 15, 19, 57–58, 62, 74, 254, 292
 - generating 70
- ATML files
 - convert from CSV 57
- ATML format 16, 62, 77, 251, 254
- auxiliary verbs 142

B

- blogs 6
- bookmark 132, 134
 - accessing 133
 - managing 133
 - saving 133
- bootinfo 43

C

- candidate file 177–178
- case studies
 - overview 242
- categories
 - add, edit, and delete 168
 - from structured content 90, 92
 - from unstructured content 90, 92
 - sort 92
- category 22, 25, 90, 123
 - association, keyword 64
 - create new 270
 - parent 64
 - Root 64
 - search 101
 - standard 22, 63
 - system 22, 63, 307
 - user defined 22, 64
- category tree 65, 90, 167, 242, 255
 - editing 168
- Category view 10, 91, 94, 97, 102, 108, 111, 116–117, 145, 242, 262, 264, 307
 - correlation values 145
 - understanding 117
- category_entry name 69
- category_tree.xml 57, 64–65
- category-tree.xml 192
- cause of accident 7
- cells
 - highlighted with pink bars 123
 - highlighted with pink shades 127, 131
- clauses 142
- column data 73
- Comma Separated Value (CSV) 15, 60, 138, 248
- Comma Separated Values (CSV) 26

- Common Analysis Structure (CAS) 17
- Complaints category 154
- COMPONENT Description 67–68, 98, 100, 232, 272
- component.adic.xml 178
- Components category 63, 145–146, 185–186, 222–223, 227–228
- COMPONENT DESCRIPTION category 116
- confidentiality
 - breach 280
- configuration 192
- configuration file
 - Data Ingestor 70
- constant noise 159
- constraint 203
 - rule 215
 - type 191
- CONTENT Category 232
- convert
 - CSV to ATML 57, 77
 - rpf to pat format 214
- convert CSV to ATML 57
- coordinate clauses and phrases 142
- correlation 4, 7, 10, 117, 127–128, 130–131, 148, 151, 240, 274
- correlation detection 26, 222–223, 239
 - setting up 230
- correlation numbers
 - 2D Map view 131
- correlation value 116–117, 142, 145, 234
 - simple set 151
- craftsman 2
- crawler 86, 303
 - full crawl 304
- cruise control 147–148, 176, 179
 - keyword 180
 - new synonym 183
 - switch 147
- CSV data
 - format 15
- CSV data file 62, 70
- CSV file 15, 61, 73–74, 138–139, 248–249
 - columns 15
 - corresponding columns 72
 - first line 16
 - multiple rows 62
- CSV files
 - convert to ATML 57
- CSV format 15, 19, 57, 61, 86, 248–249, 270, 297
 - exporting data 248
 - input data 26
 - source data 64
- CSV output 138
- csv.character.encoding 71, 75
- csv.column.index.date 74
- csv.column.index.id 74
- csv.column.names 72, 76
- csv.column.text.indexes 71, 73
- csv.date.format.list 74
- csv.date.index.title 75
- csv.row.firstindex 71, 75
- customer satisfaction 11
- customer sentiment 7
- customer-relationship management (CRM) 6

D

- data
 - delete 84
 - exporting, CSV format 248
- data export
 - to CSV files 61
- data exporting
 - into CSV format 57
- Data Ingestor 19, 21, 57, 65, 86, 251, 254
 - batch process 70
 - command 70, 77, 254
 - configuration file 15, 70
 - convert CSV to ATML 77
 - tool 70
- data ingestion 64, 86, 113
 - quick reference 86
- data type 22, 24, 65, 245, 268
 - entire category tree 24
- data_ingester_config_csv2atml.xml 70
- database
 - creation and registration 58
 - OmniFind Analytics Edition 22
- database directory 56, 58, 235
 - absolute path 80
- database resource files 58
- database_config 18, 249
- database_config.xml 57, 64, 68, 93
- database_entry name 59–60, 248, 306
- dataset 87–88, 142, 144, 166–167, 224, 231, 260
 - generalization 4
- date criteria 104
- date field 62, 74, 113

- date search 104
- decay 160
- decaying factor 229
- defect 162
- delete index 84
- delete MIML file 84
- delta analysis 9–10, 18, 23
- Delta view 94, 97, 104, 111, 125, 127, 158–159, 161, 223, 278
 - Increase Indicator Index 161
 - understanding 127
- densities 150
- detection
 - problematic behavior 7
- dictionary 78
 - apply new dictionary 291
 - custom, creating 309
 - custom, integrating 309
 - OmniFind Analytics Edition 25
 - registering OmniFind Analytics Edition dictionary with OmniFind Enterprise Edition 311
- dictionary customization 243
- Dictionary Editor 24–25, 50–51, 65, 68, 78, 97, 165–166, 168, 177, 192, 197, 249, 254, 309
 - actions 174
 - create a new category 271
 - home page 166
 - main functions 24
 - multiple instances 168
- Docs view 94, 111–112, 240
- Document Inspector 113
 - sample 114
- driving under influence (DUI) 97

E

- e-mail database
 - analysis 282
 - case study 280
 - details 281
- entity detection 17
- environment variable 39, 47, 302
- exporting data
 - CSV format 248
 - into CSV format 57
 - to CSV files 61
- expression
 - negative 196

F

- features
 - OmniFind Analytics Edition 9
- fire 146
- frequency 117, 123, 142, 145
- frequency count 119, 121
 - unexpected deviations 123
- frequency vs. correlation 145
- fters 191, 214
- fuel 148

G

- generalization
 - dataset 4
- generated CSV file
 - sample copy 139
- generating
 - ATML files 70
- generating MIML file 79
- global_config.xml 59, 90, 306
- gratitude 92

H

- HEAP_SIZE_MB 83
- heat map 151
- helping verbs 142
- highlighted cells 123, 127, 131
- highlighting 155
 - search keywords 115
- HTML 5

I

- ignition 148
- ignition switch 147, 274–275
 - short circuiting 276
- increase detection 222–223, 226
- Increase Indicator 158
- increase indicator 125–127
- Increase Indicator Index 159, 161
- index
 - building 304
 - delete 84
 - indexing MIML files 57
 - indexing quick reference 86
 - rebuilding 314
 - registering and browsing 306
- index build command 304

- Indexer 18, 21
- insight 4, 11, 18
- installation file
 - OmniFind Analytics Edition 32
- integration
 - architecture, OmniFind Analytics Edition and OmniFind Enterprise Edition 298
 - OmniFind Analytics Edition with OmniFind Enterprise Edition, high level view 27
- internal memory 43
- interval estimation 151
- item 296
- item type 296

J

- Java regex
 - interpreter 194, 200
 - processing 193
 - processor 207
 - reference 193
 - syntax 193
- Java Runtime Environment 31

K

- keyword 63
 - category association 64
 - multiple synonyms 181
- keyword search 9, 101, 187
- keywords 22, 24, 64, 123
 - multiple, search condition 102
 - remove 187

L

- language annotators 78
- language identification 17
- language processing 254
- language specific segmentation 17
- legal liability 280
- lemma 208
- lemmatization 208
- Lex 191, 209
- lex constraint 207
- Lexical item 191
- lexicon 191
- liability
 - legal 280
- linguistic analysis 8

M

- marketing strategy 6
- memory 43
- metadata 5, 298, 307
- metadata field 153, 307, 311
 - e-mail database 153
- Microsoft Excel 62
- MIML file 16, 18, 21, 57, 78–79, 83–84, 254, 293, 297–298
 - analytical indexes 57
 - delete 84
 - generating 79
 - OmniFind Analytics Edition index 314
 - reindexing 262
- mine data versus analyze data 4
- Mining Markup Language (MIML) 16, 57, 293
- multiple synonyms
 - for a keyword 181

N

- National Highway Traffic Safety Administration (NHTSA) 22, 55, 87–88, 221
 - sample database 55
- natural language
 - grammar 97
- natural language annotator 57, 63, 79, 297, 307
- natural language process 178
- natural language processing 3, 16, 18, 57, 78, 101, 113, 142, 192, 206–207, 219, 236, 247, 254, 270, 293, 298–299
 - capability 14
 - extracts noun 176
- natural language processing (NLP) 16
- natural language processing resources 78
- natural language processor
 - use 177
- negative construction
 - rules 196
- negative pattern
 - trouble 193, 197
- NetView automation 260–261
- NHSTA data 63–64
- NHTSA complaints database
 - analysis 270
 - schema 268
- NHTSA data 22–23, 91–92, 162
- NHTSA database
 - schema 65

NLP 19
noun sequence 92, 94, 223, 230
Category view 94

O

OmniFind Analytics Edition 1–2, 4, 7, 13–14, 22, 29–30, 32, 39, 55–56, 87–88, 142, 145, 165–166, 189–190, 241, 295, 297
application 20–21
architecture 15
batch program 62
command 78, 84–85
configuration 38, 48
convert CSV to ATML 57
Data Ingestor 73
database 22, 70
dictionary 25, 166, 312, 314
edition 9
environment 38, 48
index 18, 111, 270, 297
installation 30–31, 33, 43, 297, 301
installation file 32–33, 43
installation process 51
natural language processor 16–17, 90, 215
preprocessor 16, 254
server 132–133
system dictionary 97
Text Miner application 26
topology 60
OmniFind Enterprise Edition 26, 62, 86, 211, 215, 295–296
OmniFind Enterprise Edition (OEE) 26
online manual 26, 50, 53

P

parent category 64
parse 303
parser
monitoring 304
Part of Speech (POC) 151
Part Of Speech (POS) 118, 274
Part of Speech (POS) 63, 92, 191, 196, 242, 255
parts.synonym.csv 177
pattern 18, 23
pear files 300
registration 302
Phrase Constituent 92, 242, 255
phrases 142

POS constraint 197–198
preprocessing 56
preprocessor 86
prerequisites 31
problem trend 7
product message 6

Q

quick reference
data ingestion and indexing 86

R

Redbooks Web site 317
Contact us xiv
regular expression
interpreter 194
regular expression syntax 193
reliability correction 150
report 132, 134
creating 134
viewing 136
reputation
damage 280
reserved_by_system 69
resource 296
result 7
result set 5, 7, 113, 146–147, 217
Rich Pattern Format 204
Root category 64
adding 170
root cause 112, 117, 148
rpf file
convert to pat format 214
rule 23, 196
rule constraint 215
editing 204
rule creation 218
flow chart 219
Rule Editor 23, 25, 190, 192, 203–204, 209, 219
rule name 201
rule value 201
rules 78
apply new rules 291
customization 242
negative construction 196
sample, capture generic trouble 291
sample, capture negative phrases 289
sample, capture people unable to do something

- 289
- sample, capture question related information 287
- Rules Editor 78

S

- sales trend 7
- sample database 35, 45, 55, 58, 87, 122, 221
 - resource files 59
- sample document 137
- sample rules
 - capture generic trouble words 291
 - capture negative phrases 289
 - capture people unable to do something 289
 - capture question related information 287
- scientific goal 4
- search 4, 10, 101, 104
 - keyword 9, 101
 - semantic search 9
 - unstructured data 4
- search condition 18, 62, 73, 88–89, 96–97, 151, 154, 258–259, 272–274
 - append multiple keywords 102
 - buildup 242
 - system messages 262
- search engine 4
- search keywords
 - highlighting 115
- search result 4–5, 63, 89, 97–98, 101, 134, 145, 176, 201, 296
 - Category view 116
 - meaningful insight 4
- semantic search 9
- sentiment
 - customer 7
- server topology
 - multi-server 21
 - single 20
 - two 20
- source data 30, 90, 95, 167, 176, 292
 - CSV files 57
 - preprocessing steps 56
 - structured column 63
 - structured data 63
 - unstructured content 63
 - unstructured data 63
- standard item category 22, 63
- startServer (command) 38

- statistical analysis 8–10
- stemming 207
- string constraint 199
- structured content 90
- structured data 14
 - analysis 7, 9
- subcategories 93
 - display and hide 93
- synonyms 24, 181, 309
- system category 22, 63, 73, 92, 167, 307
- system topologies 20

T

- takmi 2
- TAKMI Alert 130
- TAKMI application 41
- TAKMI format 205
- TAKMI MINER 294
- TAKMI.jacl 39
- TAKMI_DIC 51
- TAKMI_HOME 40
- TAKMI_indexer_batch 82, 304–305
- TAKMI_MINER 50, 88
- TAKMI_MINER_EAR 292
- Technical Help Desk 242
 - analysis 254
 - business scenario 244
 - data 254
 - efficiency 264
- Text Analysis
 - Engine 301–302
- text analysis engines (TAE) 16
- text analytics 3–4, 142, 309
- Text Miner 9–10, 18–21, 23, 50, 53, 57, 60, 63, 83, 87–90, 141–142, 145, 151, 159, 192, 201, 206, 222–223, 230, 254–255, 262, 314–315
 - accessing 88
 - category page 217
 - Category view 116
 - deploying 88
 - Increase Indicator Index 159
 - interface 23–24, 26, 97
 - report 132
 - screen layout 89
 - single instance 90
 - user interface 105
 - various views 111
- text mining 1, 3–4, 6, 11, 140–142, 189–190

- detailed information 140
- time period 121, 124, 142
- time scale 120–121, 229
- time series 62, 73, 94, 154
- Time Series view 10, 18, 23, 97, 104, 108–109, 111, 119, 153, 223, 238, 276
 - understanding 120
- tokens 22
- Tools view 134
- Top view 111
- Topic view 10, 18, 23, 94, 97, 104, 110–111, 121, 123, 153–155, 158, 161, 277
 - understanding 123
- Topicality index 152, 154
- trend analysis 9–10, 18, 23
- trouble negative pattern 193, 197

U

- UIMA annotators 27, 297
 - uploading and associating 300
- unfavorable 92
- unstructured content 5, 63–64, 90, 148
 - language annotators 64
- unstructured data 14, 30, 63, 145, 163, 190, 255–256
 - analysis 7, 9
 - search 4
- Unstructured Information Management Architecture (UIMA) 9, 11, 14, 16–17
- unstructured text content 167
- URL
 - launching Dictionary Editor 261
 - launching Text Miner application 255
- use case 1, 5, 236, 239
- user defined category 22, 64
- user interface 23–24, 49–50, 105, 155, 159, 174, 179
 - numeric value 155
- UTF-8 format 59

V

- variation scale 158
- verification
 - application 50
 - installation 51
- views
 - 2D Map view, Text Miner 128, 130–131
 - Category view, Text Miner 116–117

- Delta view, Text Miner 125, 127
- Docs view, Text Miner 112
- Time Series view, Text Miner 119–120
- Top view, Text Miner 111
- Topic view, Text Miner 121, 123, 153
- voice of customers 6, 87, 92, 148–149, 242, 255, 265

W

- Web browser 50–51, 166, 168
- WebSphere Application Server 30–33, 43, 84, 291, 293
 - directory 41
 - entry 38
 - environment 30
 - installation root directory 49
 - normal installation 50
 - start 38
- weighted keyword 160

X

- XML 5
- XML descriptor file 17
- XML element 69
- XML file 59, 68, 115, 249
- XML version 59, 68

Introducing OmniFind Analytics Edition: Customizing for Text Analytics



(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Introducing OmniFind Analytics Edition: Customizing for Text Analytics



Installing and configuring

Customizing libraries and rules

Case studies

IBM OmniFind Analytics Edition uses information in unstructured and structured content to improve decision making. It provides unstructured information analysis, which can improve customer service levels, create cross-sell capabilities, and provide early problem detection.

IBM OmniFind Analytics Edition gives business users the tools to gain valuable insights, facilitating better business decisions, by exploring and using key information assets in the enterprise. It also provides multiple ways to explore and analyze information, and it delivers sophisticated entity extraction capabilities when working with unstructured content. It is based on the open Unstructured Information Management Architecture (UIMA) standard.

This IBM Redbooks publication will help you understand the power and capability of IBM OmniFind Analytics Edition Version 8.4. The book provides instructions on installing and configuring IBM OmniFind Analytics Edition. It explains how to use it for text mining, and how to work with Dictionary Editor and Rule Editor to customize the application for better analysis and discovery. The book also discusses how to set up the Alerting System to automatically watch for increased unusual activities. The integration with IBM OmniFind Enterprise Edition (OEE) is also covered.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks