Enhancing Business Intelligence with unstructured data

Alexander Lang, Maria Mera Ortiz, Stefan Abraham

Advanced Analytics Development
IBM Research and Development Germany
Schoenaicher Str 220
71032 Boeblingen
alexlang@de.ibm.com
mmera@de.ibm.com
stefana@de.ibm.com

Abstract: Quality early warning and proactive customer churn detection are two examples of applications that can benefit from insights gained from unstructured text data. The term "Unstructured Business Intelligence" describes methods and tools that enable data warehouse applications to use unstructured information. This article introduces the key components of Unstructured Business Intelligence and describes its realization within IBM InfoSphere Warehouse Enterprise Edition.

1 Unstructured Business Intelligence

1.1 Motivation

In a recent TDWI survey [Ru07], data management professionals were asked *Which types of data and source systems feed your data warehouse three years from now?* The respondents expected a huge increase in unstructured data as a source of warehouse information in the next three years. This included email, call center transcripts, documents from content management systems, and extranet content from forums or blogs.

This article will show how text analysis technology can transform this unstructured, textual data into meaningful pieces of information that can be used within Business Intelligence applications. Unstructured data can improve the quality of existing BI analytics, or, in some cases, is the key enabler for new types of insights. Typical business scenarios include:

- *Improve Quality Early Warning*: Internal problem reports, customer email or call center transcripts can yield valid information about emerging product problems. Today, companies try to capture these insights using a fixed set of categories within "problem taxonomies". Such taxonomies typically suffer from granularity problems: if they contain only high-level categories, they can't capture the actual reason for a problem. However, if they try to capture all possible problems, they become too unwieldy to use for front-line workers, who just stick to the categories they know (especially in a high-stress environment such as a call center). Thus, the actual reason for a defect is often buried within technician comments or call center logs. As a result, a company may detect that there is a problem with a certain product, but doesn't know which part causes the problem, and therefore can't take the right action: deciding on a product recall, or checking other products that use the offending part. On the contrary, Figure 1 shows a sample report showing correlated terms extracted from customer complaints for a certain car model, which provides direct insight in likely problem spots.

- *Reduce customer churn*: Companies in the telecommunication sector already have elaborate predictive analytic models for customer churn, based on structured data. However, once the customer's unhappiness with a certain service shows up in the structured data (for example, a decline in the number of long-distance calls made), it may already be too late. By analyzing each customer contact with the company, be it email or call center records, a company can earlier detect angry, unhappy customers, or customers that explicitly reference a competitor, and include that into their churn model. This allows for taking action at the first sign of customer discontent.

- *Reputation management*: Blogs, news articles and consumer portals increasingly affect customer's buying decisions, especially for consumer goods. Analyzing this extranet content helps to answer questions like *How do people talk about my company and my products – compared to my competition?* or *What companies are associated with "cool" technology?* – for example, hybrid technology in the automotive domain. This can serve as an "External Quality Early Warning" system, as not every unhappy customer bothers to write to the company, but whose forum entries may turn away quite a few prospective customers.

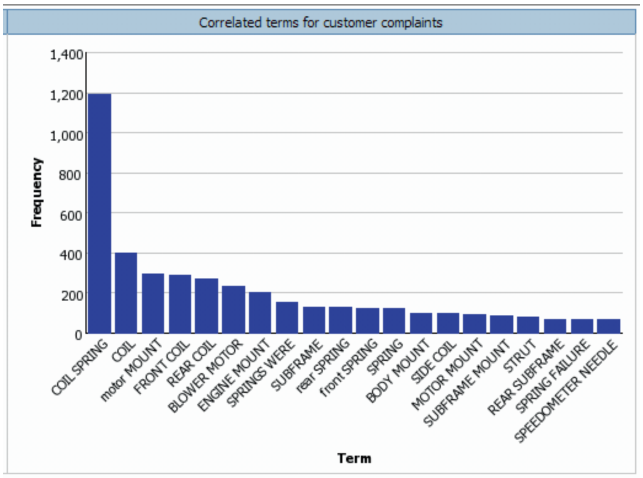


Figure 1: Analysis of automotive customer complaints

In all of the above scenarios, text is the main type of unstructured data. There may be the need to analyze semi-structured text, i.e. text containing markup elements like XML, or other data types of audio and video. However, we see that the bulk of content that is relevant for today's applications comes as free-form text from technician notes, customer comments through CRM applications or email, or relevant snippets from news services. Thus, the remainder of this article will focus on this type of text.

1.2 Definition: Unstructured Business Intelligence

Using text data for Business Intelligence comprises three steps:

1. Introduce dedicated text analysis components into the ETL flows that feed the data warehouse. These text analysis components extract information from the unstructured data, such as product names, product codes, indicators for problems or expressions of customer sentiment.

2. Enrich existing warehouse schemas and cube definitions with additional columns or dimensions for the extracted information. This requires that the text analysis technology delivers results that can be readily integrated into a relational schema.

3. Use the existing BI infrastructure on the enriched warehouse. Ideally, reporting, data mining and predictive analytics should not need to care that the additional information was originally extracted from an unstructured source.

We use the term *Unstructured Business Intelligence* (or *Unstructured BI*) to describe methods and tools that enable data warehouse applications to use unstructured information. This includes

- Text analysis components that extract information with sufficiently high quality to be used by automatic downstream processing.
- Approaches to create efficient relational schemas for text analysis results that enable both reporting and data mining.
- Analytics components that can combine structured and previously unstructured data to yield better results and more insight.

Unstructured Business Intelligence includes approaches that are often described as *Text Mining*. However, we think that text mining has too close a connotation with data mining, and leaves out important aspects such as data preparation, relational schema building and the use of text in "traditional" business intelligence reports.

1.3 Challenges

Using textual data in Business Intelligence presents some challenges to "typical" data warehouse infrastructure. Typical requirements include:

- *Dealing with misspellings and acronyms*: In a project with a Telecommunications customer, we started with an Excel sheet that contained the names of products, call tariffs and contract options. However, this sheet didn't include the variants used by call center employees. Adding these company-specific acronyms, for example *NOK* for *Nokia,* more than doubled the number of concepts that got extracted from call center records. Moreover, robust text analysis technology is required to deal with frequent misspellings (e.g. *Nioka*).
- *Advanced data cleansing:* Especially dealing with extranet content requires data cleansing and preprocessing capabilities that are typically beyond the capabilities of "traditional" cleansing products. For example, analyzing hotel reviews from sites such as tripadvisor [Tr08] requires to focus only on the actual customer reviews, not on the advertisements or the description provided by the hotel itself. Increasingly, this data cleansing becomes a "make or buy" decision, with companies such as kapow technologies [Ka08] offering screen scraping tools and content feed services that provide such targeted information.
- *Confidence values*: Text analysis sometimes lacks the context to decide on ambiguous terms. For example, the acronym *WPS* may refer to either *IBM WebSphere Portal Server* or *IBM WebSphere Process Server*. Some text analysis components are able to attach confidence values to the extracted information. However, there are currently few BI algorithms that make use of this information.

- *Agile text analysis configuration:* Text analysis technology typically has to be adapted to each domain, enterprise, and business scenario. Product names must be imported from Excel sheets or product databases and enriched with company-specific variants. Regular expression rules need to be created for product codes, taxonomies need to be developed for relevant topics. The key item for the success of an *Unstructured BI* project is the buy-in of domain experts and business users. Only they are able to provide the insights necessary for configuration. The IT department may take over some of the tasks (e.g., writing regular expressions), but only the business users know the relevant "business lingo". Hence, a potentially large user group must be able to influence the configuration in an agile manner. Otherwise, the resources will become outdated, yielding inferior results, and eventually, user rejection of the whole solution.

2 Text analysis: from text to structure

2.1 Information Extraction

Information Extraction is an area of natural language processing that is concerned with extracting concepts (so called entities) and relationships between these concepts from unstructured text. These results lend themselves for storing within a relational database, and are thus more relevant to *Unstructured BI* than technologies such as document clustering or classification. Relevant information extraction tasks are:

- *Named Entity Recognition (NER)*: extract person or place names, monetary expressions, problem indicators

- *Relationship Detection*: detect relationships, based on named entities, such as *part X causes problem Y*

- *Co-reference resolution*: identify expressions across a document that refer to the same entity, such as the hotel *Best Western* in *I liked my stay at the **Best Western**. **It** has very bright rooms. **The hotel** also features…*

2.2 List-based and rule-based named entity recognition

Named Entity Recognition tasks are of varying complexity: *The Grosvenor Inn did not meet my expectations* contains both a hotel name and a negative customer sentiment. Whereas the hotel name can be detected by a match against a fixed list of hotels, the correct analysis of the negative sentiment requires some grammatical understanding to correctly pick up the negation.

One approach to Named Entity Recognition is *list-based extraction*. This is advisable for entities that can be exhaustively listed, such as employee names (taken, for example, from the company LDAP), or product names and their attributes. Some domains, such as Life Science already have "official" domain vocabulary, e.g. SnoMed CT [Sn07].

One advantage of list-based extraction is that it often comes from "trusted" sources. That means its creation and maintenance can be automated to a certain degree, e.g. by doing a "batch update" every time new products are introduced. Moreover, the extraction results are immediately plausible to the end user: the machine detects exactly the terms that are in the list. As mentioned earlier, the most labor-intensive part in using list-based extraction is the enhancement of the list with company-specific variants and acronyms.

Some types of entities, such as telephone numbers or monetary expressions can't be listed exhaustively. For these entities, *rule-based extraction* is the proper approach. These may either be regular expression rules over characters, or higher-level rules, such as regular expressions over previously detected entities. One advantage of rules is generalization: one rule may cover a large range of entities. Another advantage is that rules can take the document context into account. This is crucial for tasks like sentiment detection, where a negation word such as "not" flips the sentiment of a whole sentence.

The key challenge for rules is their complexity: users need help to create and maintain rules. For many people, writing regular expressions is as obscure as programming in Java – but these are often the people with the appropriate domain knowledge. Hence, configuration tooling is necessary that hides the intricacies of linguistics and rule languages from non-technical users. For complex tasks such as sentiment detection, even that is not enough. Creating the syntactical rules necessary for negation and other sentiment-affecting phenomena will overwhelm non-Linguists. In these cases, a combined approach is in order: the solution contains pre-packaged grammar rules, which are not meant to be changed by business users, but lets users modify the list of positive and negative terms, or the products and product features that should be related to an extracted sentiment.

2.3 Using BI analytics for Information extraction tasks

In “traditional” computational linguistics, relationship detection and co-reference resolution is often approached through elaborate rule sets. However, such rule sets are often very specific to a particular domain, or assume a formal writing style that often does not exist in the type of text relevant for *Unstructured BI*. In our experience, it is often beneficial to accept some loss in extraction precision for higher robustness against “real-world” text. This can be achieved by relying on co-occurrence heuristics and statistics, and lends itself easily to the machinery available within existing BI components. Relationship detection between parts and problems can be done by extracting parts and problems through NER, and applying data mining, namely, Association rule mining [AIS93], on the result to find parts and problems that occur within the same document in a statistically significant amount. Co-reference resolution can be done by associating all product features or customer sentiments within a certain document snippet to the concept (e.g., the hotel) found within that snippet. The right snippet length may vary from one sentence up to a paragraph, depending on the type of document, e.g. a consumer forum vs. a website with product reviews.

2.4 Decision factors for choosing information extraction technology

In general, choosing the right set of information extraction components depends on:

- *Business scenario:* How complex are the entities that need to be detected? For example, a purely quantitative analysis of product mentions in a call center doesn’t require as elaborate processing as detecting dissatisfied customers within the same documents.

- *Configuration effort:* As stated in chapter 1, text analysis needs to be adapted to an enterprise, with business users playing a key role in the process. Do the expected business improvements merit the (ongoing) configuration effort for the desired technology?

- *Focused vs. explorative solution approach:* In a *focused solution* approach, the goal of the text analysis configuration is to extract the relevant concepts with high precision. This requires more effort for creation and testing the configuration, but allows the downstream BI analysis users to pose specific questions like: *Show the top 10 car parts that occur in repair reports for automobile make X.* In the *explorative solution* approach, text analysis detects terms in a very broad fashion, and doesn't attempt to match them against a certain concept, for example, by detecting all combinations of an adjective, followed by a noun. This requires more elaborate statistical analysis or data mining post processing to detect statistically significant concepts and relationships. Still, the results may show irrelevant terms, and the user can't do a concept-driven drill down like in focused solutions. However, an explorative solution is more suited for "open-ended" questions like *What are the causes for failure X?* where the root cause may be a car part, a certain tool used, or even a certain repair shop. Moreover, such an approach keeps the text analysis configuration effort to a minimum, and is able to pick up emerging concepts on the fly, without the need for a human to add them to a list of concepts.

3 Unstructured Information Management Architecture (UIMA)

3.1 Motivation

Scenarios such as company reputation management typically require a set of text analysis components to work together. For example, customer satisfaction analysis in the hospitality sector requires both list-based extraction to detect hotel names and hotel amenities, and rule-based extraction to detect customer sentiment. Chances are that the expression *Best Western Shanghai* will be tagged correctly as a hotel name, but the term *best* may trigger a positive sentiment rule as well, so that *Best Western* is tagged as a customer sentiment. To spot these inconsistencies, and apply subsequent cleansing steps such as *remove all customer sentiments that are totally included within a concept,* the text analysis components must represent their results in a common way. Moreover, a framework should manage the task to orchestrate different analysis components (which may come from different sources), rather than forcing solution builders to re-implement these capabilities.These requirements were the driving force behind the Unstructured Information Management Architecture (UIMA). UIMA is a software architecture that supports the rapid development, integration and deployment of unstructured information management technologies, including, but not limited to, text analysis [UI04]. UIMA started as an IBM initiative, but has now gone open source, and is enhanced through the Apache community [UI08]. The *UIMA Working Group* is a network of scientific organizations and companies that use UIMA in research projects and professional development.

3.2 Core concepts

Analysis Engines are the central building blocks within UIMA. An analysis engine contains one or more *annotators,* each implementing one specific text analysis functionality, or other analysis engines. This recursive packaging allows to build complex analysis engines out of simpler ones.

Analysis Engines work off a common data structure, the *Common Analysis Structure (CAS).* Each analysis engine stores its results within the CAS as typed feature structures. *Annotations* are a special subclass of feature structures. They contain a specific start and end position within a document and lend themselves easily for specifying information extraction results, e.g. as a *Hotel* annotation that covers positions 5 to 26 in the document *The Best Western Shanghai welcomes you.* This open data representation allows individual teams to develop an analysis step that is "best of breed" in what it's doing, and which builds on results delivered through prior annotators rather than recomputing them. Figure 2 shows several annotators working together for named entity recognition, grammatical parsing and relationship detection. Note that the *Relationship Annotator* can detect relationships without looking at the actual document text, but by analyzing pre-existing concepts and grammar.

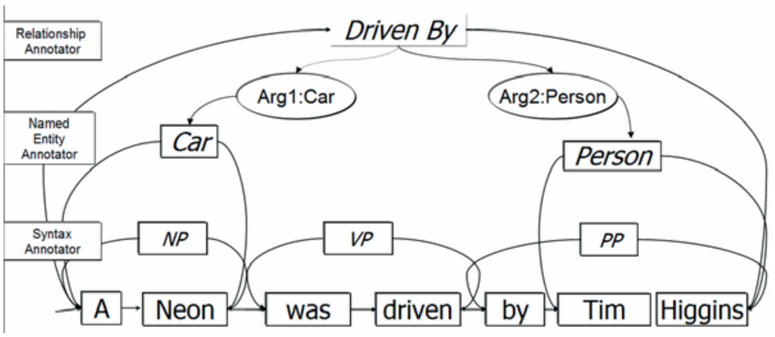


Figure 2: UIMA analysis process

3.3 Text analysis configuration workflow

UIMA aims to ease the integration of text analysis components, but does not provide text analysis configuration capabilities suitable for business users. Figure 3 describes this configuration as an iterative, ongoing process, with the following distinct phases:

Explore Data: The goal of this step is to understand "what's in the data", and to understand whether the textual data can actually help with the business problem. Tools that support the domain expert in this step are components that suggest potentially relevant terms from the documents, based on document statistics. Another way to explore the data are text search queries. The results may contain "signal terms" that can be used in regular expressions to improve their quality, or, by using fuzzy search, to find frequent typos and variants to include into dictionaries.

Create configuration: In this step, the domain expert decides on the extraction approach to use (e.g., rule-based vs. list-based) and creates the appropriate resources. This step requires easy-to-use editors, which are able to re-use the insights gained in the explore data step.

Evaluate: In this step, the domain expert examines the effectiveness of the rules and lists on sample, real-life data. This requires tools to run the analysis and to compare results, which is important to gauge the impact of configuration changes.

Deploy: In this step, the created rules and resources are deployed to the target system. Note that this is the only step that does not require any business users, but should be done through the IT department.

Monitor: In this step, feedback obtained in the production system over time must be applied to the resources to ensure that the configuration remains up to date. This is especially relevant in the focused solution approach. Monitoring the extraction recall through BI reports (e.g., *How many products have been detected on average per call center transcript*) can help detect concept drift: emerging concepts that are not reflected in lists or rules, and thus decrease the overall recall.

The *domain expert* in the process must be familiar with the business goals of the unstructured BI initiative, and with the documents that are to be used. He must work with front-line employees creating these documents to capture company-specific jargon.

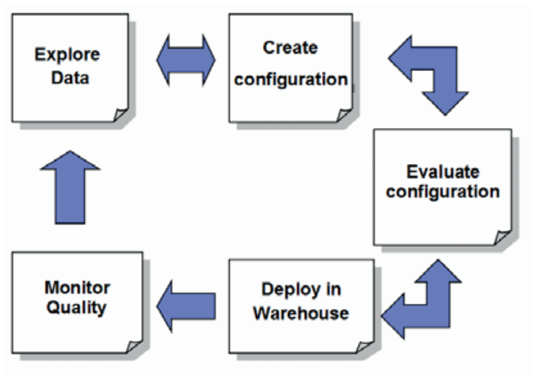


Figure 3: Text analysis configuration workflow

4 Unstructured Business Intelligence with InfoSphere Warehouse

4.1 InfoSphere Warehouse Enterprise Edition

InfoSphere Warehouse Enterprise Edition [In07] is a complete solution to design and build an enterprise data warehouse. In addition to “traditional” ETL and OLAP cube modeling, it provides integrated data mining and visualization. The data mining capabilities cover both discovery analytics through association analysis, sequential patterns and clustering and predictive analytics through various classification and regression algorithms.

Building up a suitable warehouse through ETL steps or creating and visualizing a data mining model for business insight is done graphically by defining *flows* within the *Design Studio,* an Eclipse-based design environment. These flows contain operators that either represent SQL commands or higher-level tasks such as text analysis. They are then deployed to a flow execution environment within IBM WebSphere Application Server. Depending on the operator, the actual tasks are pushed down to the underlying database, or executed as Java components with the application server.

4.2 InfoSphere Warehouse enhancements for unstructured BI

InfoSphere Warehouse 9.5 introduced key capabilities for Unstructured BI. First of all, text analytics became a first class citizen within ETL flows by introducing three new flow operators:

- *Dictionary lookup* runs a UIMA analysis engine for list-based extraction. It performs linguistic preprocessing of the text for more than 20 languages to make sure that plural forms or inflections of concepts are also detected. This way, a list entry for "wiring harness" also detects "wiring harnesses" and "wire harness", and “Fertighaus” also detects “Fertighäuser” (German for *pre-fabricated home*). Note that a simple “stemming” approach to plural detection, which simply cuts off word endings, would have problems matching the resulting stem *Fertighäus-* back to *Fertighaus.*

- *Regular expression* runs a UIMA analysis engine for rule-based extraction. It is capable of detecting concepts based on regular expressions over characters. Expressiveness and syntax conform to the java.uti.regex.Pattern class of Java 5 [Ja08]

- *Text analyzer* runs a custom UIMA analysis engine. This analysis engine may have been custom-built for a certain solution, or can be an analysis engine provided by other IBM products or the Apache community.

Above operators process the text contained in a CHAR, VARCHAR or CLOB column of a database table, and produce one or more output columns.

To create dictionaries and regular expression rules, the Design Studio was enhanced with dedicated editors that hide the underlying linguistic processing, or the actual rule language from the domain expert. A taxonomy editor allows assigning the extracted concepts to higher-level categories. This is especially relevant when using text analysis results in an OLAP context, as it allows describing additional aggregation levels over the extracted concepts. For example, a dictionary with IT terms like Java and Windows enables a report like “What are the TOP 10 IT terms?”. However, creating a taxonomy with the paths *IT-Skills -> Programming Languages -> Java* and *IT-Skills -> Operating systems -> Windows,* allows for reports like *What are the Top 10 IT-Skills?* or *What are the Top 10 programming languages?*. By separating the taxonomy from the dictionary, different applications can impose their “view” on the extracted concepts, without requiring any changes to the text analysis itself.

The configured analysis engines run as Java components within the application server. Several analysis engines can run in parallel on the same database table to enhance document throughput.

5 Example scenario: Quality Early Warning

The Office of Defects Investigation is an office within the National Highway Traffic Safety Administration (NHTSA) in the United States. It maintains a web site where consumers can file complaints about vehicles, equipment (e.g., motorcycle helmets), child restraints or tires [Od08a]. The contents of the complaint database are publicly available and contain 511,068 unique complaint records. Each record contains 46 structured fields, plus an unstructured complaint description, which can contain up to 2048 characters [Od08b]. As Figure 4 shows, the complaint description has some “typical” characteristics for unstructured database content: it is in all-uppercase format, and does not exceed one paragraph in length.

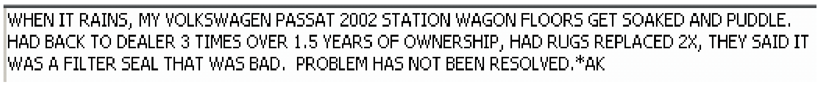

WHEN IT RAINS, MY VOLKSWAGEN PASSAT 2002 STATION WAGON FLOORS GET SOAKED AND PUDDLE. HAD BACK TO DEALER 3 TIMES OVER 1.5 YEARS OF OWNERSHIP, HAD RUGS REPLACED 2X, THEY SAID IT WAS A FILTER SEAL THAT WAS BAD. PROBLEM HAS NOT BEEN RESOLVED.*AK

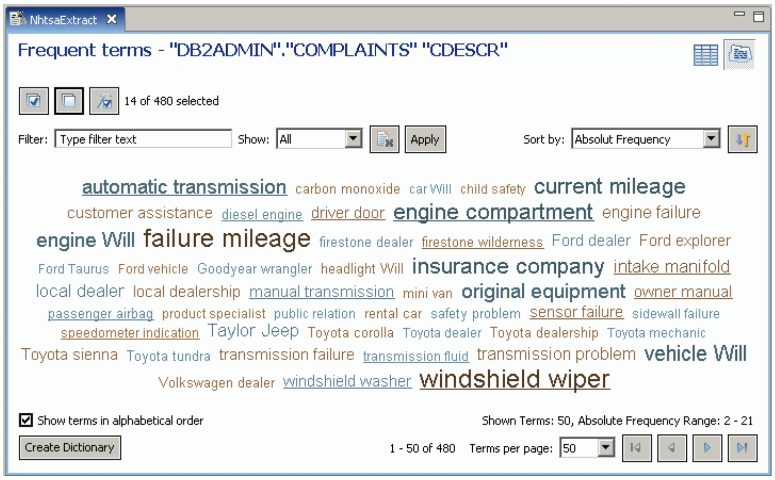
Figure 4: Sample complaint description

While the structured fields contain descriptive information about the complaint, such as whether the vehicle was involved in a crash, or the number of people injured, they contain little information that point to the cause of the problem. The complaint may be categorized, but categories like *TIRES*, *AIR BAGS* and *OTHER* are not granular enough to describe the problem adequately.

This example scenario shows how InfoSphere Warehouse can extract relevant information from the complaint descriptions, which allows business analysts to investigate potential causes of problems for certain vehicles. In this example, the Quality Early Warning application is built as a set of reports within IBM Cognos 8 BI server. As the results of both text analysis and subsequent data mining are stored in relational tables, the application could be implemented in any BI tool, or custom application.

5.1 Text Analysis Configuration

In the *Explore Data* phase, we use InfoSphere Warehouse to find frequent terms within the complaint descriptions. These terms are extracted based on grammatical patterns such as *Noun-Noun* or *Adjective-Noun*. The domain expert selects the appropriate patterns, and InfoSphere Warehouse shows the results in a list or a cloud view, as in [Fig X]. In our example, we focus on grammatical patterns that match car parts, such as *Noun* (e.g., engine) or *Noun-Noun* (e.g. windshield wiper). As the underlined terms in figure 5 show, the results of frequent term analysis with above patterns would provide a good starting point for creating a dictionary with car parts. A domain expert could select relevant terms, add them to a dictionary, and enrich the entries with additional variants.


NhtsaExtract

Frequent terms - "DB2ADMIN"."COMPLAINTS" "CDESCR"

14 of 480 selected

Filter: Type filter text Show: All Apply Sort by: Absolut Frequency

automatic transmission carbon monoxide car Will child safety current mileage customer assistance diesel engine driver door engine compartment engine failure engine Will failure mileage firestone dealer firestone wilderness Ford dealer Ford explorer Ford Taurus Ford vehicle Goodyear wrangler headlight Will insurance company intake manifold local dealer local dealership manual transmission mini van original equipment owner manual passenger airbag product specialist public relation rental car safety problem sensor failure sidewall failure speedometer indication Taylor Jeep Toyota corolla Toyota dealer Toyota dealership Toyota mechanic Toyota sienna Toyota tundra transmission failure transmission fluid transmission problem vehicle Will Volkswagen dealer windshield washer windshield wiper

☑ Show terms in alphabetical order

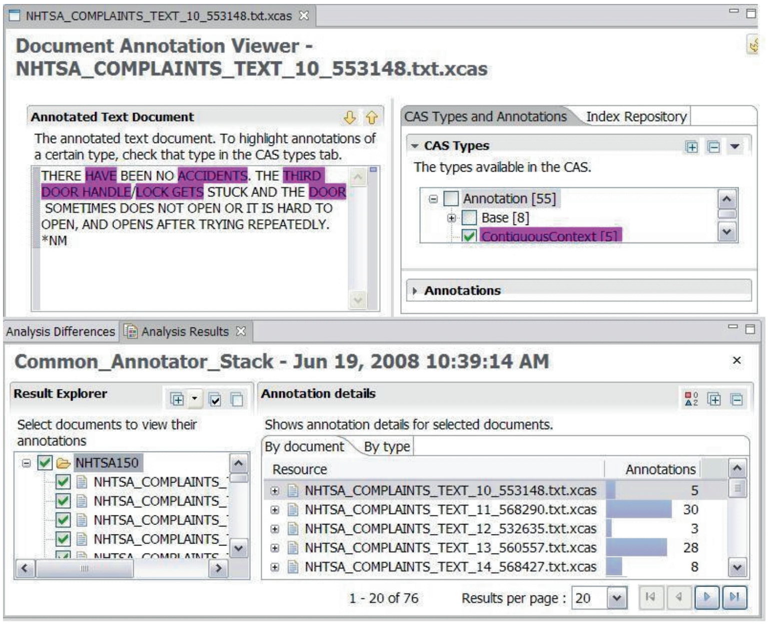
Shown Terms: 50, Absolute Frequency Range: 2 - 21

Create Dictionary 1 - 50 of 480 Terms per page: 50

Figure 5: Frequent noun-noun combinations

However, in the example scenario, we chose to take an explorative solution approach. Rather than creating a dictionary specifically with car parts, we add all extracted noun-noun combinations and single noun combinations to a dictionary. Hence, the Create Configuration phase merely consists of a single mouse click on the *Create Dictionary* button shown in figure 5. This approach yields a sizeable list of entries, but the dictionary lookup operator compiles this list into an efficient binary representation, which provides a lookup time linear to the length of the lookup term, not linear to the length of the list, as in a naïve implementation. This allows to process several documents per second, even if the list grows to hundreds of thousands of entries.

In an explorative solution approach, the *evaluate* phase is used to check whether the dictionary contains entries from all relevant grammatical patterns. Depending on the document content, it may be beneficial to create additional dictionaries containing verbs that signify actions, or adjectives that describe some product features or circumstances. Figure 6 shows the result of dictionary lookup on a sample complaint. The tab *Analysis Results* contains an aggregated results view, which shows how many terms got extracted for each document. This allows the domain expert to gauge the recall of the current configuration, and to specifically check documents that contain zero or few results whether they contain terms that are not captured by the current configuration. On the other hand, documents with an exceptionally high number of terms may point to "stopwords" that are contained in the dictionary.


NHTSA_COMPLAINTS_TEXT_10_553148.txt.xcas

Document Annotation Viewer - NHTSA_COMPLAINTS_TEXT_10_553148.txt.xcas

Annotated Text Document

The annotated text document. To highlight annotations of a certain type, check that type in the CAS types tab.

THERE HAVE BEEN NO ACCIDENTS. THE THIRD DOOR HANDLE/LOCK GETS STUCK AND THE DOOR SOMETIMES DOES NOT OPEN OR IT IS HARD TO OPEN, AND OPENS AFTER TRYING REPEATEDLY. *NM

CAS Types and Annotations | Index Repository

CAS Types

The types available in the CAS.

- Annotation [55]
  - Base [8]
  - ✓ ContiguousContext [5]

Annotations

Analysis Differences | Analysis Results

Common_Annotator_Stack - Jun 19, 2008 10:39:14 AM

Result Explorer

Select documents to view their annotations

- ✓ NHTSA150
  - ✓ NHTSA_COMPLAINTS_
  - ✓ NHTSA_COMPLAINTS_
  - ✓ NHTSA_COMPLAINTS_
  - ✓ NHTSA_COMPLAINTS_

Annotation details

Shows annotation details for selected documents.

By document | By type

Resource	Annotations
NHTSA_COMPLAINTS_TEXT_10_553148.txt.xcas	5
NHTSA_COMPLAINTS_TEXT_11_568290.txt.xcas	30
NHTSA_COMPLAINTS_TEXT_12_532635.txt.xcas	3
NHTSA_COMPLAINTS_TEXT_13_560557.txt.xcas	28
NHTSA_COMPLAINTS_TEXT_14_568427.txt.xcas	8

1 - 20 of 76 Results per page : 20

Figure 6: Text analysis result overview

In the *Deploy* phase, the created dictionary is used within a Dictionary Lookup operator of an ETL flow (figure 7). The text column CDESRC from the complaints database is connected to the input port of the dictionary lookup operator. This operator is configured to use the dictionary created in the evaluate step, as seen on the *properties* tab. The result is a table that contains one record per extracted term. The column CMPLID from the original table is routed through the operator as a join key with the original complaint.

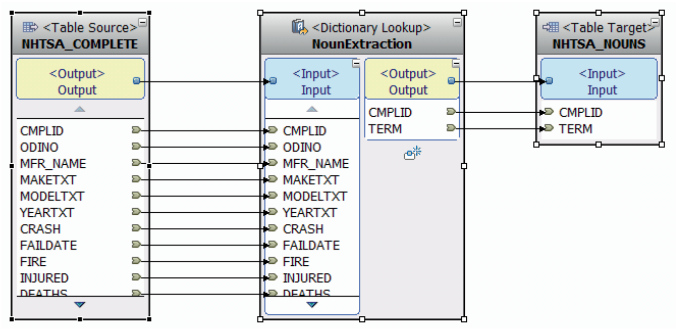


Figure 7: Text analysis ETL flow

The result table can now be used within a Business Intelligence application such as IBM Cognos 8 BI server. For the example, we chose car manufacturers, car models and the term list as dimensions for the data cube. This allows the domain expert to start with an overview how many complaints exist for each manufacturer. She can then drill-down to the car models of a certain manufacturer to see models that have a high number of complaints. Up to this path, the information is derived solely from the existing structured information. A further drilldown to a certain car model yields the insights gained from analyzing the car complaints. Figure 8 contains a report that shows the most frequently occurring terms for the car model Ford Taurus. However, showing only the most frequent terms for a particular model is not enough: entries such as CAR occur across all complaints, and don't yield insight. This underlines our statement made in chapter 2: the explorative solution approach needs analytical post processing of the results to detect terms that are correlated with the subject under discussion, in this case, the Ford Taurus.

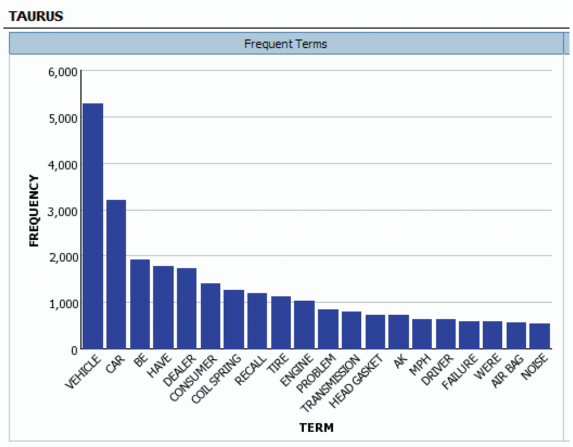


Figure 8: Most frequent terms in complaint descriptions for Ford Taurus

5.2 Combining text analysis and data mining

To find correlated instead of frequent terms in our example, we use Associations analysis that is part of InfoSphere Warehouse Data Mining. The goal of association analysis is to find items that are consistently associated with each other in a statistically significant way [AIS93]. Association analysis is typically used for *Market-Basket-Analysis*, which analyzes purchase transactions to discover combinations of goods that are often purchased together.

In our scenario, the goal of the association mining function is to find terms that are highly correlated with a certain car model. The results are rules of the form [rule body] ==> [rule head], e.g. [ventilator oil] ==> [TAURUS]. Figure 9 shows a flow that uses the text analysis results to build an association rule model. This model is built within all models of a certain car make (e.g. Ford), and attempts to find terms that are highly correlated with a certain model (e.g., Taurus). The latter is done by applying a rule filter that only accepts rules that contain Taurus in the rule head.

The extractor operator shown in figure 9 stores the detected rules back into a relational table. This way, the results can be visualized with "traditional" BI, and both administrators and users of BI applications don't need to be aware that the results were detected using data mining.

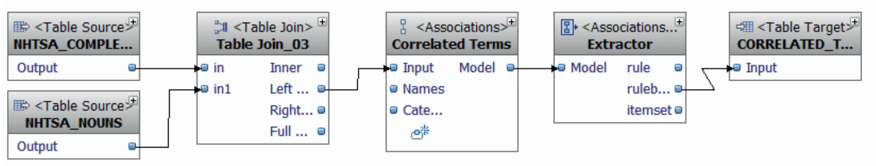


Figure 9: Using association analysis to detect correlated terms

Figure 1 contains a report that shows the most correlated terms for Ford Taurus, as detected by association analysis. The terms are much more specific – "stopwords" such as car and vehicle do not occur anymore. Many of the results focus around the *coil spring*, with both *front coil* [spring] and *rear coil* [spring] being affected. The second problem spot appears to be certain *mounts*, such as the *motor mount* or the *subframe mount*. The domain expert can now drill down into the actual documents that contain a certain term to understand whether there indeed is a problem.

The goal of this example scenario is not to take the business user completely out of the loop, but to keep down the information that (s)he has to consume to a manageable size: instead of reading through all 14000 complaints for the Ford Taurus, the business analyst can focus on the 80 that contain the term *body mount*. If the resulting document set is still too large (e.g. for *coil spring*, which still yields 1200 documents), the domain expert can focus on a sample set, or do a further drill-down on the term. This drill-down action triggers the creation of a new association model for the documents that contain, say, coil spring. The base set used to derive the association rules is now the set of all Taurus documents, not the set of all Ford documents. This way, the system can guide the user along relevant, correlated terms to specific "problem spots".

6 Summary

The data warehouse will experience a paradigm shift in the coming years: structured data remains important, but the ability to harness unstructured data will become key to support and enhance new business intelligence applications. This article has introduced the term *Unstructured Business Intelligence* to describe the methods and tools necessary to integrate unstructured data into the warehouse. In the context of a Quality Early Warning scenario, the article showed how IBM InfoSphere Warehouse enables building Unstructured BI applications. Unstructured BI includes text analysis components that perform information extraction, and tools and processes that allow the agile configuration of these components. But the work doesn't stop there: advanced data cleansing approaches and downstream analytic steps that can combine structured and previously unstructured data are equally relevant to deliver a complete Unstructured BI solution.

7 References

[AIS93] Agrawal R., Imielinski T, Swami A: Mining Associations between Sets of Items in Massive Databases. In Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data, 1993; pp. 207-216

[In07] Official product page:http://www.ibm.com/software/data/db2/dwe/enterprise.html

[Ja08] J2SE 5 documentation: http://java.sun.com/j2se/1.5.0/docs/api/java/util/regex/Pattern.html

[Ka08] Kapow Technologies, http://www.kapowtech.com

[Od08a] ODI home page: http://www-odi.nhtsa.dot.gov/index.cfm

[Od08b] Database schema is available at http://www-odi.nhtsa.dot.gov/downloads/folders/Complaints/CMPL.txt

[Ru07] Russom, P.: BI Search and Text Analytics, TDWI Best Practices Report, 2007; pp. 9-11

[Sn07] http://www.ihtsdo.org/our-standards

[Tr08] Tripadvisor Travel Web Site; http://www.tripadvisor.com

[UI04] Unstructured Information Management, Special Issue of IBM Systems Journal, Vol.43,No. 3, 2004

[UI08] Website of the Apache incubator project: http://incubator.apache.org/uima/