Glossary extraction and utilization in the information search and delivery system for IBM Technical Support

by L. Kozakov
Y. Park
T. Fin
Y. Drissi
Y. Doganata
T. Cofino

In this paper we describe the practical aspects of extracting and using a glossary for a selected technical domain. We first describe the existing glossary extraction process, as applied to general corpora, and examine its shortcomings in the technical support domain. Then we propose a number of enhancements to it, including focusing the glossary on a selected domain context, providing support for multidomain glossaries, and importing domain-specific dictionaries. We apply our focused-glossary approach to the IBM Technical Support corpus and incorporate resulting glossaries within the information search and delivery system used by IBM Technical Support. We demonstrate the effectiveness of our approach by evaluating the quality of keywords and terms extracted from sample documents with the help of these glossaries.

As information technology (IT) advances, the number of products released and their associated documents increase at a rapid pace. Technical authors, and in particular authors of documents for product support, often use terms and words that are not found in general-purpose dictionaries. A situation may arise in which the meaning given to technical terms varies within the technical community.

Terminology frequently changes with the introduction of new products to the market. The terminology database in a technical support organization is perhaps one of the most frequently updated databases of this kind. The IBM Technical Support knowledge base, for example, contains specifications, problem descriptions, proposed solutions, and updates on thousands of hardware and software products.

Glossaries can alleviate this problem. Glossaries help build a language common to people who search for information and people who author documents, thus increasing the effectiveness of search and retrieval systems. Because glossaries change rapidly and because of their large size, generating glossaries manually is costly. We describe in this paper the results of our investigation into automated glossary extraction. We also describe how we used an improved glossary extraction process to build and deploy a number of glossaries within the IBM Technical Support system used by customers. The business justification for building glossaries is to increase customer satisfaction when they use the IBM Technical Support Web site.

Technical documents pertaining to IBM products and services are processed, indexed, and stored in a master repository, known as the electronic support knowledge base (eSKB) or the knowledge repository, which contains about a million documents in several languages. We have used eSKB as the corpus for our glossary extraction process, which integrates a number of tools and components into a complete solution.

The effectiveness of glossary extraction depends strongly on domain-specific resources, such as dictionaries, and also on the rules that generate labels or error codes, like APAR (authorized program analysis report) numbers or SQL (structured query language) errors. The glossary extraction processes, which are normally trained on general corpora like TREC (Text Retrieval Conference)[1] and do not take into account a specific domain, such as technical support, produce less useful glossaries for technical support applications. We found that domain-focused glossary extraction, where the term weights depend on document context, improves the effectiveness of the glossary.

In this paper we show several ways to improve the usefulness of the glossary and to make it more effective and robust for technical-support applications. To demonstrate the value of our approach we implemented Keyword Analyzer (KWA), an application that identifies salient terms in a document by using weighted terms from the glossary.

The rest of the paper is structured as follows. In the next section we present an overview of the architecture of the information search and retrieval system used by IBM Technical Support. Then, we summarize the approach to glossary extraction from Reference 2, which we will use as our starting point. In the following section we describe our implementation of the automated glossary extraction process for the technical support corpus and discuss the results we obtained. We observe that implementing the glossary extraction process without considering the specifics of the domain may lead to some erroneous results, and consequently, we present suggestions for improvement. Next we introduce KWA, the application we use to evaluate the effectiveness of our approach. We then propose the concept of a domain-focused glossary, in which glossary items are selected and ranked based on context, and we show some quantitative results from our tests. In this section we also discuss a possible application of the domain-focused glossary: the improvement of document-relevancy ranking in corporate search systems. We summarize our work in the concluding section.

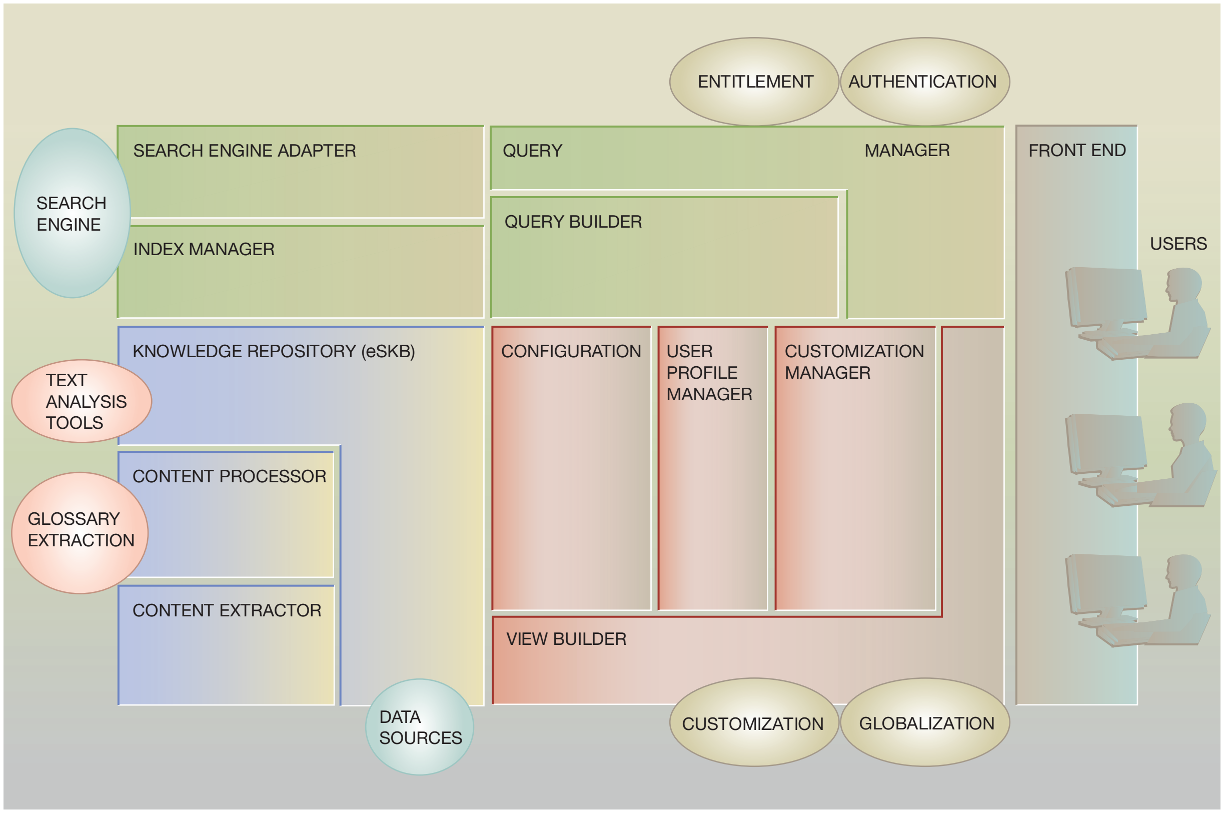
Overview of the IBM Technical Support Enablement Architecture

The IBM Technical Support Enablement Architecture, whose implementation is nicknamed *dBlue*, is an advanced information search and delivery architecture for the Web-based system used by IBM Technical Support.[3] One of the goals of this system is to help customers find the desired information among the 2.5 million Web pages stored on the system. The dBlue system, which integrates effective technologies in storing, searching, and retrieving information, provides a set of user-oriented support services used by all IBM support sites.

The architecture connects three important types of elements from the information search world—information sources, search engines, and end users (see Figure 1). This is done through a set of components called the Knowledge Builder, which includes a content creation layer (blue blocks), a search management layer (green blocks) and a presentation management layer (red blocks). Information sources are any structured and unstructured data sources such as document repositories, DB2 and Lotus Notes databases, Web sites, and so forth. The first challenge of the architecture was to institute a consistent structure for content creation because the huge amount of support content that already existed was not well suited for searching. Then, of course, both existing and new content had to be migrated to this structure. The second challenge was determining how to store this information in a way that was scalable and flexible. The third challenge was how to retrieve it dynamically and efficiently. The main blocks of this architecture are shown in Figure 1. Content is extracted from the information sources using the Content Extractor and mapped to a unified XML (eXtensible Markup Language) schema. Then it is processed by the Content Processor and stored into the eSKB. The search management layer enables the connection between the Knowledge Builder and search engines. The Query Manager and Query Builder are responsible for processing search queries, collecting query-related parameters from the configuration management layer, and building the final search query. The presentation management layer provides several levels of customization, based on country, organizational unit, and individual user profiles. The View Builder constructs a customized view of search hitlists and documents. When a user requests a view of a specific document, this request is processed by the View Builder, which accesses the eSKB to retrieve the document content, and builds a coherent document view.

At the content creation layer the architecture enables the use of text analysis tools, technologies and resources, including the Technical Support glossary, for enhancing document content and improving the

Figure 1 IBM Technical Support enablement architecture

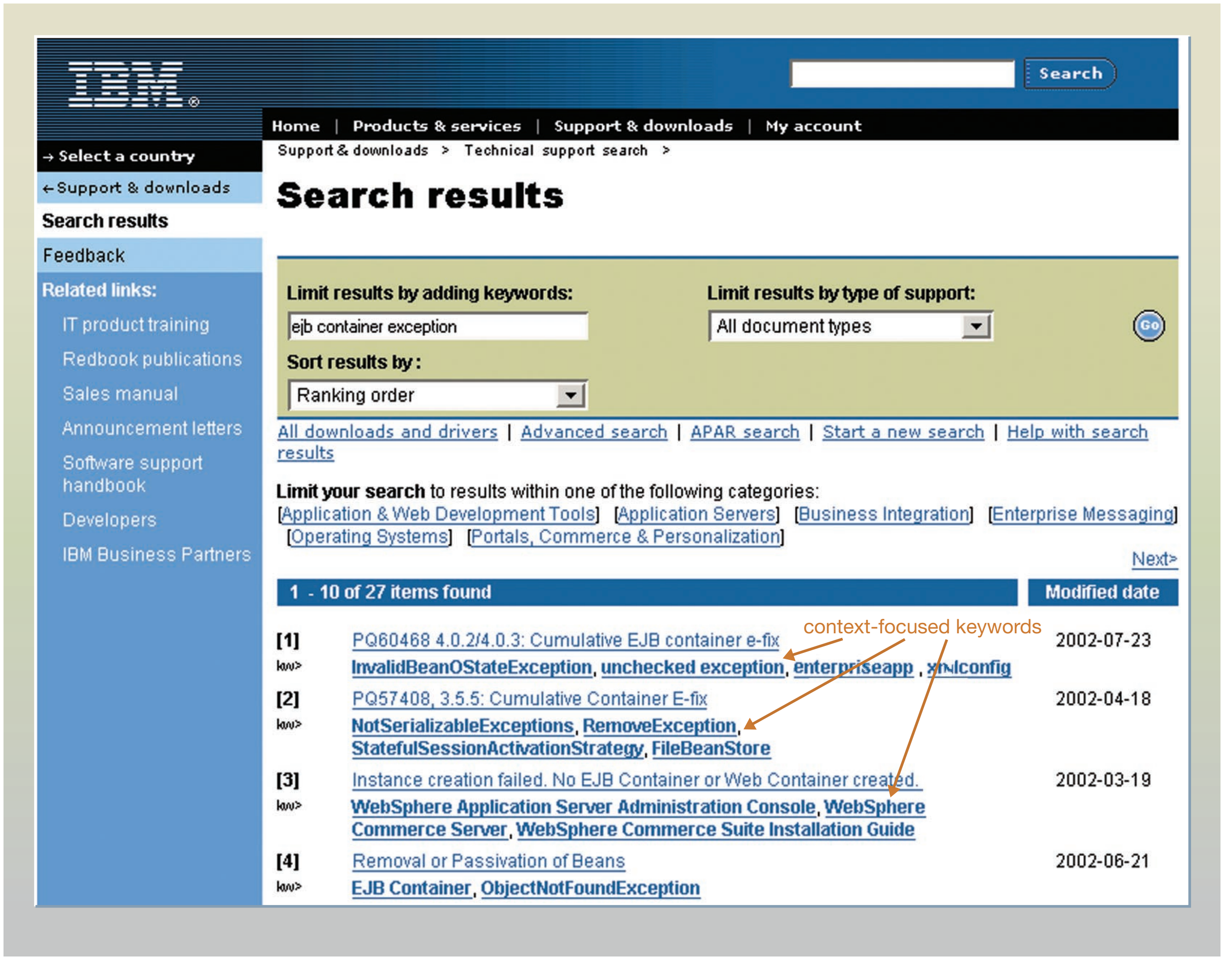


search experience. The Technical Support glossary, for instance, may be directly used as the vocabulary of domain-specific terms for spell-checking by various content management and search applications. The text analysis tools include keyword analysis and content summarization, briefly described in this section. The text analysis tools are also responsible for keeping the Technical Support glossary synchronized with the Technical Support corpus. Additional information on the dBlue system can be found in Reference 4.

The purpose of the keyword analysis utility is to identify domain-specific salient single-word or multiword terms in the document. The identified keywords are stored in the repository together with the corresponding documents and used as document metadata for improving the search experience. A content summarization utility is used to create meaningful summaries of the stored documents. These summaries, which are stored in the repository along with the documents, are also displayed on search-results pages, thus further improving the search experience. We describe next the keyword analysis and content summarization techniques and their use in the dBlue system. For more details about the content summarization technique see Reference 5. Evaluation of different types of summarization and discussion of a related usability study with regard to the IBM Technical Support corpus can be found in Reference 6.

To identify domain-specific salient terms (keywords), each incoming document is processed by KWA. This utility is based on the TALENT[7] text analysis engine (TAE) configured to work with the Technical Support glossary. TALENT (Text Analysis and Language Engineering Technology) is a general suite of text analysis tools developed at the IBM Thomas J. Watson Research Center that recognizes significant objects in text (such as names, terms, relations, parts of speech and abbreviations) and annotates documents with this information (for more details see

Figure 2 Display of search results with keywords



Search

Home | Products & services | Support & downloads | My account

→ Select a country

Support & downloads > Technical support search >

←Support & downloads

Search results

Search results

Feedback

Related links:

IT product training

Redbook publications

Sales manual

Announcement letters

Software support handbook

Developers

IBM Business Partners

Limit results by adding keywords: ejb container exception

Limit results by type of support: All document types

Sort results by: Ranking order

Go

All downloads and drivers | Advanced search | APAR search | Start a new search | Help with search results

Limit your search to results within one of the following categories:
[Application & Web Development Tools] [Application Servers] [Business Integration] [Enterprise Messaging] [Operating Systems] [Portals, Commerce & Personalization]

Next>

1 - 10 of 27 items found		Modified date
[1]	PQ60468 4.0.2/4.0.3: Cumulative EJB container e-fix	2002-07-23
kw>	InvalidBeanOStateException, unchecked exception, enterpriseapp, xmlconfig	
[2]	PQ57408, 3.5.5: Cumulative Container E-fix	2002-04-18
kw>	NotSerializableExceptions, RemoveException, StatefulSessionActivationStrategy, FileBeanStore	
[3]	Instance creation failed. No EJB Container or Web Container created.	2002-03-19
kw>	WebSphere Application Server Administration Console, WebSphere Commerce Server, WebSphere Commerce Suite Installation Guide	
[4]	Removal or Passivation of Beans	2002-06-21
kw>	EJB Container, ObjectNotFoundException	

Reference 7). The TALENT TAE analyzes the content of the input document, identifies single- and multi-word terms along with their variants (using the Technical Support glossary as the reference vocabulary), and returns the keywords and their variants together with the domain-specific confidence level of each in the Technical Support glossary.

In a similar fashion, KWA returns a sorted list of identified domain-specific keywords along with their variants to the content processing layer. After the keywords are stored in the knowledge repository as meta-data, they are indexed by the search engine indexer as the document meta-data, along with the document content itself, to enable keyword-based search as well as some other search enhancements. The keywords can also be used to change the relevancy score of the document: if the search-query terms match one of the document keywords, the score of the document may increase, reflecting increased likelihood this document is relevant for the domain associated with the given search terms. Moreover, we can use the keywords to improve the search experience by displaying them along with the document title on the search result page, as shown in Figure 2.

Aside from improving the search experience, KWA can also be used for improving the document authoring process. Authors can use KWA interactively to select appropriate keywords for the documents they are writing. This is especially useful in situations where the authors have limited time for writing or interpreting text, such as customer service representatives entering the particulars of a service call in the system.

Because Technical Support document abstracts or summaries are rarely supplied by content authors, dBlue supports the automated generation of document summaries through the use of the content summarization utility. This utility is based upon the TALENT TAE working with the Technical Support glossary. The TALENT TAE content summarizer implements a well-known sentence extraction model[8,9] that analyzes lexical cohesion factors in the source text.[5] Sentence extraction is driven by the notion of salience—the resulting summary is constructed by identifying and extracting the most salient sentences in the document. The salience score of the sentence is defined partly from the salience of the Technical Support glossary terms in it, and partly from its position in the document structure and the salience of surrounding sentences. The document summary, created by the content summarization utility is stored in the repository as meta-data, and is displayed on a search-results page along with the document title. In a usability study it was shown that accurate automatic content summarization may help customers identify the relevant documents among search results, thus improving customer satisfaction.[6]

Overview of the glossary-extraction process

Glossary items are single-word or multiword phrases representing the domain concepts in a given document collection. A domain-specific glossary encapsulates important conceptual material in that domain. For practical purposes, glossary items name and describe the domain concepts in a way that can be exploited by applications.

In this section, we briefly describe the glossary-extraction system, *GlossEx*,[2] that we use in our implementation. GlossEx, which is designed to automatically extract glossary items from a large unstructured document collection, has the following main components: glossary-item identification, glossary-item acquisition, filtering, glossary-item aggregation, and ranking (see Figure 3). Although in Figure 3 the components are shown connected to a central pipe in which data flow from left to right, a more accurate illustration would have each component receive its input from the upstream components and in turn feed its own results to the downstream component. We point out that such a pipeline architecture allows for easy insertion of new components or replacement of an existing component. For instance, users can easily replace the part-of-speech (POS) tagger with a new POS tagger, or add more filters according to the domain or the style of the documents.

Glossary-item identification. The identification of candidate glossary items includes a number of steps: lexical look-up, morphological analysis, POS tagging, and noun phrase recognition. Although GlossEx can generate any type of word and users can select the POS of target glossary items, in most cases a glossary item is either a noun phrase or a verb. For verbs, we consider only non-auxiliary verbs and take their base forms as candidate glossary items. For noun phrases, we derive the structure of noun phrases using formula (1), which is based on the study by Justeson and Katz[10] and domain experts' analysis of the documents.

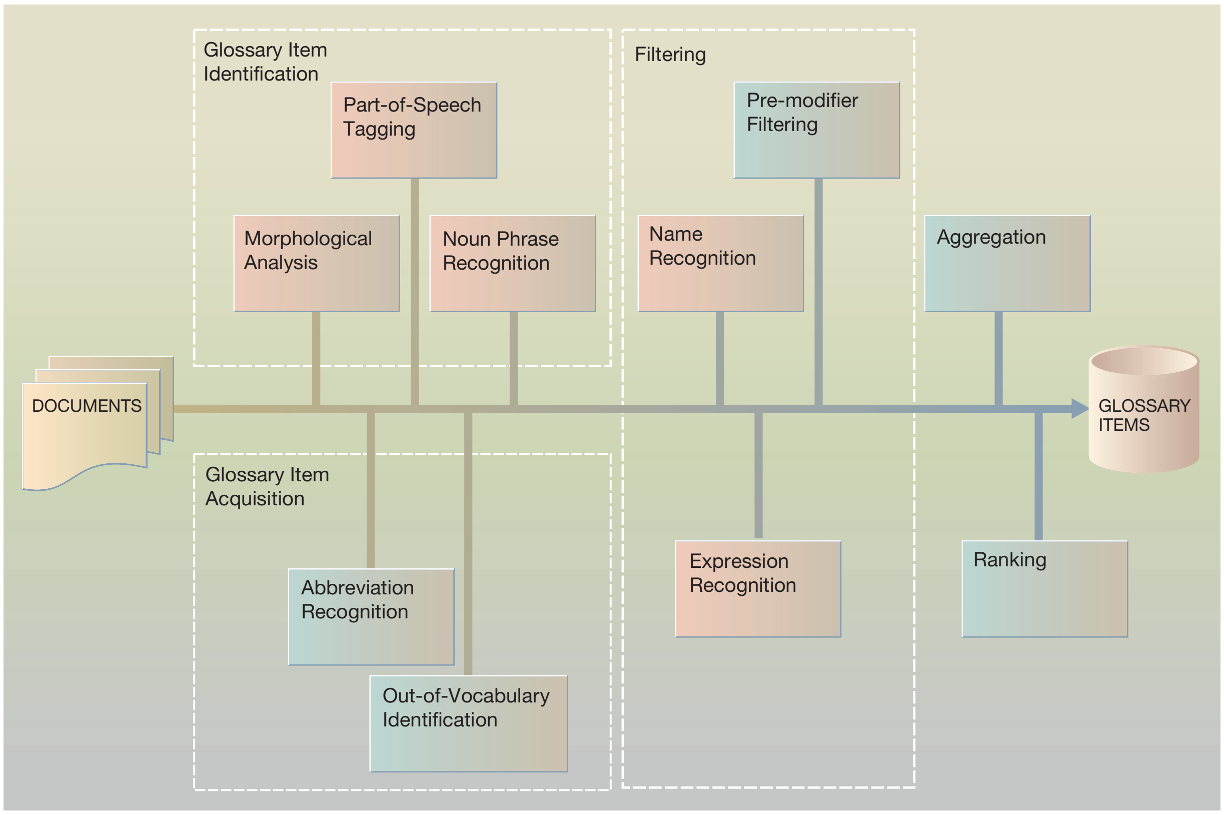
$$\begin{aligned}
&\text{Noun} = NN \mid NP \mid NNS \mid NPS \\
&\text{AdjectivalModifier} = JJ \cdot (CC \cdot JJ)^* \\
&\text{Glossary item} = (DT \cdot (VBG \mid VBN)^+) \\
&\quad \cdot (\text{Noun} \mid \text{AdjectivalModifier})^* \cdot \text{Noun}. \qquad (1)
\end{aligned}$$

Here *DT* denotes a determiner, *VBG* and *VBN* denote participle forms of verbs, *JJ* denotes an adjective, *CC* denotes a conjunction, *NN* and *NNS* represent singular and plural common nouns, and *NP* and *NPS* denote singular and plural proper nouns. We also use the following symbols: '$\mid$' denotes "logical OR," ' $\cdot$ ' denotes "sequence," '*' denotes "zero or more times," '+' denotes "one or more times."

The recognition of glossary items based on the grammar sketched in formula (1) is realized as a cascade of finite-state (FS) transducers.[11,12] The reasons for using FS technology include reusability of grammars, adaptability to different domains and applications, and portability across languages. This last consideration is essential for a number of applications, where glossary extraction needs to be carried out over multilingual text collections. In principle, it is easier to adapt an implementation to a different language if phrasal patterns are specified as linguistic abstractions and interpreted by a language-independent engine.

Glossary-item acquisition. Documents, and especially technical documents, contain many abbreviated forms and out-of-vocabulary (OOV) words (that is, words that do not appear in a given dictionary). *Remanufacturability*, *oxidizability*, and *airbreathing* are

Figure 3 The pipeline architecture of GlossEx



examples of OOV words. Examples of abbreviations (and their definitions) in technical documents are: 4H (four-wheel drive high), DOHC (double overhead cam) and NVH (noise, vibration, and harshness).

The proper recognition of abbreviations and OOV words is very important for understanding technical documents and for extracting information from them. Thus, GlossEx includes components for finding abbreviations and their definitions and also for finding OOV words in documents. We refer readers to Reference 13 for abbreviation processing and to Reference 14 for OOV word processing.

Term filtering. The steps to identify and acquire glossary items generate a large number of candidate terms, not all worth considering. We conduct two kinds of term filtering: discarding unimportant glossary items and removing non-domain modifiers from noun phrases.

Examples of unimportant glossary items are terms consisting of more than six words, person names, arbitrary strings including special characters or digits, addresses, and URLs (uniform resource locators). We apply several heuristics to identify these expressions and then discard them. For person or location names we use a precompiled "names" dictionary from TALENT.[7]

A more important filtering is *pre-modifier* filtering (a pre-modifier precedes the term described). Many pre-modifiers in noun phrases, even in domain-specific noun phrases, act as general-purpose modifiers rather than representing domain-specific information. For instance, a pre-modifier *remote* in glossary item *remote server* is domain-specific, but *new* in *new server* is not viewed as such.

The easiest way for filtering non-domain pre-modifiers might be to keep a "stop-word" list and remove all pre-modifiers in the stop-word list from candidate glossary items. However, some modifiers are domain-specific in one domain but general in others. Thus, we automatically decide whether a pre-modifier should be filtered based on the domain-specificity ($\mathcal{D}$) of the pre-modifier and the association ($\mathcal{A}$) of the pre-modifier with the noun it modifies. The domain-specificity of a pre-modifier a is computed by relative probability of the occurrences of the word in a domain corpus d and in a general corpus g; that is, $\mathcal{D}(a) = p_d(a)/p_g(a)$. The association of the pre-modifier with the head noun (n) is calculated by the conditional probability of the head noun and the modifier; that is $\mathcal{A}(n, a) = p(n|a)$.

Glossary-item aggregation. The same concept may appear in text in a number of different variations, such as misspellings or abbreviations. We attempt to identify all conceptually identical expressions of a candidate glossary item and aggregate them into one glossary item, so that they can be treated by applications as one.

GlossEx currently identifies and aggregates inflectional variants, orthographic variants, compounding variants, misspellings, and abbreviations. We select one of the forms as the *canonical form* and make the other forms its variants. The aggregation step also combines the frequencies of the different forms so that glossary items with many variant occurrences may be assigned higher confidence values.

- Inflectional variants: singular-plural forms and different tenses (human-performance criterion and human performance-criteria)
- Orthographic variants: glossary items with special characters such as hyphens or dashes (audio/visual input and audio-visual input)
- Compounding variants: compounding form and lexicalized form (passenger airbag and passenger air bag)
- Misspelling variants: correct spelling and misspelling or alternative spelling (accelarator and accelerator, nitroglycerine and nitroglycerin)
- Abbreviations: abbreviated form and full form (R1H and radial first harmonic)

Note that GlossEx currently does not perform deep semantic processing, and thus it cannot identify synonyms nor handle polysemous glossary items. Instead, we provide a GUI (graphical user interface) tool for users to manually add or aggregate synonymous glossary items for their applications.

Glossary-item ranking and selection. Having obtained candidate glossary items, we rank them before selecting the final set. We decide the goodness of each term based on how much an item is related to the given domain, its *domain specificity*, and the degree of association of all words in the item's canonical form (hereafter called *term cohesion*). The *confidence* of a term T, $C(T)$, is defined by equation (2).

$$C(T) = a \cdot TD(T) + \beta \cdot TC(T). \tag{2}$$

Here TD is term domain specificity, TC is term cohesion, and a and β ($a + \beta = 1$) are constant values that determine the relative contributions of TD and TC.

The degree of domain specificity. If an item is used more often in a domain-specific document than in other documents, it likely is a domain-specific term. We evaluate the domain specificity of a word based on the probability of occurrence of the word in the given domain-specific text versus that in a general corpus. We define the domain specificity of a multiword term as the average of the domain specificity of the words in the term as shown in equation (3).

$$TD(T) = \frac{\sum_{W_i \in T} \log \frac{P_d(W_i)}{P_c(W_i)}}{|T|}. \tag{3}$$

Here $|T|$ is the number of words in term T, $P_d(\mathrm{W}_i)$ is the probability of word W_i in a domain-specific document, and $P_c(\mathrm{W}_i)$ is the probability of word W_i in a general document collection. The probabilities are estimated by the occurrence frequencies normalized by the size of the corpus.

The degree of term cohesion. We describe here the method for computing the cohesion of multiword terms.[2] The goal is to measure the association of an arbitrary n-gram ($n > 1$), and to give higher values to terms having high co-occurrence frequencies. This method involves a generalization of the Dice coefficient[15] so that it satisfies the two goals as given in equation (4). The measure is proportional to the co-occurrence frequency and the length of the term.

$$TC(T) = \frac{|T| \times \log_{10} f(T) \times f(T)}{\sum_{W_i \in T} f(W_i)}. \tag{4}$$

Here $|T|$ is the number of words in term T, $f(T)$ is the frequency of term T, and $f(w_i)$ is the frequency of word w_i. Equation 4 produces much higher val-

Table 1 Sample glossary entries

Canonical Form	Attributes C - confidence TD - domain specificity	Variant Forms

BEA WebLogic Server	C = 2.913107
	TD = 7.175857
		BEA WebLogic Servers

Enterprise Java Bean	C = 3.079965
	TD = 7.653201
		Enterprise Java Beans;
		EJB**

Java 2 Platform, Enterprise Edition	C = 2.786597
	TD = 6.966474
		J2EE**

Java class	C = 3.268571
	TD = 8.045130
		Java classes;
		Classes

Java Runtime Environment	C = 3.226994
	TD = 7.750770
		JRE;
		JREs

Module Dependency	C = 2.537534
	TD = 6.153712
		Module Dependencies

SQL statement	C = 0.124499
	TD = 0.116062
		SQL statements

Warning	C = 0.656059
	TD = 1.597206
		Warnings

WebSphere Developer Domain	C = 3.795610
	TD = 7.118393
		WSDD

Web tool	C = 3.035567
	TD = 7.488574
		Web tools;
		Web tooling

ues for single-word terms than multiword terms because the association of a single-word term only depends on its frequency. Thus, we reduce the scale of association of single-word terms by taking only a fraction of the value (for example, 10 percent).

Table 1 shows sample glossary entries extracted from a collection of 2042 documents in the WebSphere* Studio Application Developer category. The table includes canonical forms of glossary entries, as well as the major attributes and different variant forms. The table shows that terms that have more relevancy to a category have higher confidence and domain-specificity levels than less relevant terms.

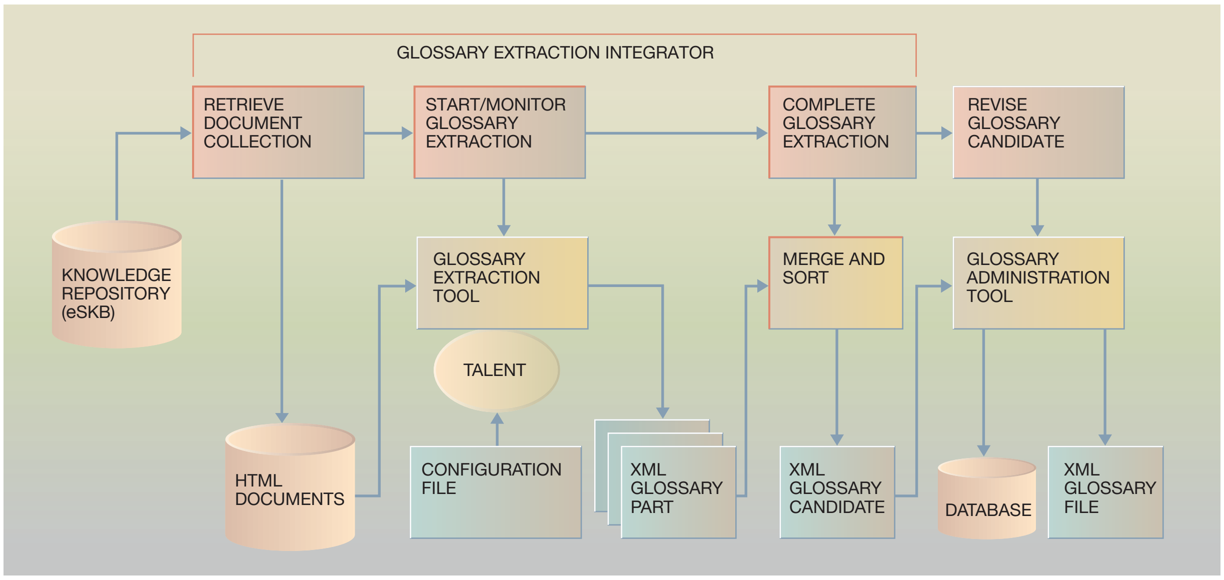
Deploying the Technical Support glossary-extraction process

In this section we discuss how to integrate the glossary extraction process into the information search and delivery system for IBM Technical Support.

The glossary-extraction process for the Technical Support corpus. Extracting a glossary from the corpus of technical support documents is more than just processing all the files in a given directory by using a glossary extractor. First, before the glossary extractor starts processing a document, the document content is retrieved from eSKB and presented in an appropriate format. Second, the glossary-extraction controller monitors the glossary-extraction process to ensure its robustness, stability, and performance. If the throughput of the process significantly degrades after a number of documents has been processed, the controller interrupts the glossary-extraction process, saves results to a partial results file for glossary candidate terms, and restarts the process with the next document in the queue. Third, after the entire document collection has been processed, the partial glossary candidate files are merged into a single file. The merging operation tabulates different variants of the same glossary term that appears in different partial files, and then recalculates all statistical attributes of each glossary term, taking into account the size of the entire document collection. These functions are performed by the Glossary Extraction Integrator, an application we developed for building the Technical Support glossary and whose architecture is illustrated in Figure 4. The same application is also used for updating the Technical Support glossary whenever the repository contains a significant number of new documents.

After the glossary candidate file is produced, it is reviewed by a glossary administrator, using a GUI-based glossary administration tool. This tool facilitates all commonly performed operations on the set of glossary terms, such as browsing, sorting, and removing or updating terms. Then, the administrator selects

Figure 4 Glossary extraction integrator


GLOSSARY EXTRACTION INTEGRATOR
RETRIEVE DOCUMENT COLLECTION
START/MONITOR GLOSSARY EXTRACTION
COMPLETE GLOSSARY EXTRACTION
REVISE GLOSSARY CANDIDATE
KNOWLEDGE REPOSITORY (eSKB)
GLOSSARY EXTRACTION TOOL
MERGE AND SORT
GLOSSARY ADMINISTRATION TOOL
TALENT
HTML DOCUMENTS
CONFIGURATION FILE
XML GLOSSARY PART
XML GLOSSARY CANDIDATE
DATABASE
XML GLOSSARY FILE

the terms that are included in the final glossary file by "approving" these terms. The terms that should not be included in the glossary, are marked as "rejected." Other terms are assigned the status "pending" until they are either approved or rejected by the glossary administrator at a future glossary update.

Field experience. In this section, we describe our experience with the deployment and use of a glossary extracted from a part of the Technical Support corpus. We also include a number of observations regarding the glossary extraction problems along with suggestions for further improvement.

The collection of Technical Support documents that we used to create the initial version of the Technical Support glossary comprised approximately 240000 items with a cumulative size of about 1 GB. About 60 percent of the documents had author-assigned categories according to the Technical Support taxonomy. The taxonomy contains 31 top-level nodes, of which 13 nodes are of type 'hardware' and 18 nodes of type 'software'. Although the taxonomy has more than 50000 nodes, there are many categories that are not represented in the document collection. Figure 5A shows the distribution of the documents within 25 non-empty top-level categories and their subcategories. *Hardware* documents account for 22 percent of the entire collection, whereas *software* documents account for the remaining 78 percent. As shown in Figure 5A, most of the *hardware* documents belong to a single top-level category (and its subcategories), whereas most of the *software* documents are distributed within six top-level categories.

Figure 5B shows the distribution of *unique* glossary entries (i.e., entries that appear only in documents that belong to one category and its subcategories) within the same 25 top-level categories. The total number of glossary entries extracted from the documents in the 13 *hardware* categories is about 100000; whereas, for the 18 *software* categories this number is about 250000. The percentage of unique entries is very high—more than 90 percent of all the entries appear in only one category. In addition, most of the identified entries contain multiword terms—more than 93 percent in both *hardware* and *software* categories.

Simple reviewing of Technical Support glossary statistics indicates that the number of unique entries is much higher than expected. Although each *hardware* or *software* category may have its own set of specific terms, we anticipated the total number of specific terms to be limited. For instance, total number of terms in the IBM terminology database[16] is about 18000, and thus we expected that most of the identified terms would appear in more than one category.

Figure 5 Technical Support corpus and glossary statistics

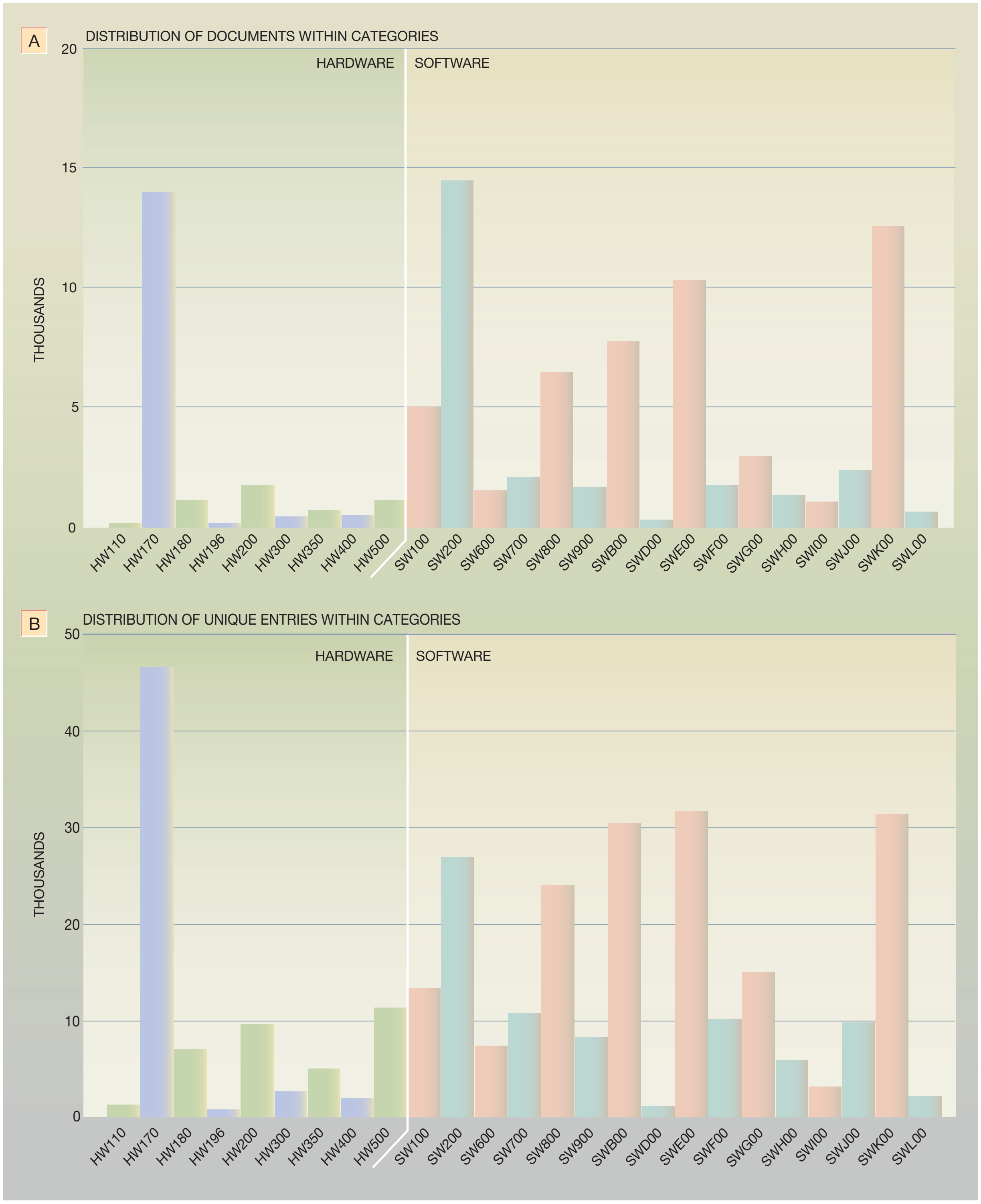


Table 2 Some GlossEx problems and suggestions for improvement

Problem/Example	Analysis	Suggestion
There are recurring situations when POS tagging does not work properly. Example: "*Change Web Resource Collection* name." *Change* was labeled as a noun when it should have been a verb. As the result, the whole sequence of words was misidentified as a multiword term.	In each example the problematic word could be labeled as a noun or a verb. POS tagger chose the wrong label because the problematic word was the first word in the sentence, and the next several words were capitalized nouns.	Take into account special case of "instructional sentences," which usually start with a verb. Modify rule-based POS disambiguation.
There are recurring situations when corporate product names are not recognized correctly. Examples: 1. *Aventail Connect*** was not recognized as a multiword term (corporate product name). 2. *WebSphere Application Server Advanced Edition Single Server* was not recognized as a multiword term (corporate product name).	Some corporate product names may contain words that could be labeled as nouns or verbs. In our case, GlossEx could not properly identify a multiword product name because the second word was labeled as a verb. Some corporate product names may exceed the default maximum multiword window used by the GlossEx utility. In these cases, GlossEx fails to identify a multiword product name.	Use comprehensive dictionary of corporate product names. Perform dictionary lookup to identify product names.
Sometimes the term cohesion parameter does not reflect the real multiword term cohesion. Example: *Borland Enterprise Server VisiBroker*** *Edition for Java* was not recognized as a multiword term.	Some multiword terms may consist of words that have significantly different frequencies in the document collection. In our example, both *Borland* and *VisiBroker* have a frequency of 2, while *Java* has a frequency of 125. In this case, the formula for term cohesion does not produce correct results, and the most frequent word may be excluded from the multiword term. Specialized dictionary lookup may not help in all possible cases.	If the words in a possible multiword term have significantly different frequencies in the document collection, formula 4 for term cohesion may not work. The term cohesion, in this case, may be calculated based on traditional analysis of co-occurrences.
There were recurring situations when domain-specific abbreviations were misidentified. Example: *stdio* was identified as a misspelling of *studio*.	Domain-specific abbreviations may be hard to identify when general rules for abbreviations are applied.	Use comprehensive dictionary of domain-specific abbreviations. Perform dictionary lookup to identify domain-specific abbreviations.

Another important observation is that the proportion of multiword terms among all the identified terms is far higher than expected. It is not comparable, for instance, to the corresponding proportion in the IBM terminology database, in which multiword terms account for 70 percent of all the *software* terms.

Analyzing these glossary statistics, we observe that the GlossEx utility has problems in recognizing multiword terms. Table 2 summarizes the observed failures of the glossary extraction process, the possible causes, and suggested solutions. The first three entries in Table 2 address different cases of multiword term recognition. The last entry in the table addresses the recognition of domain-specific abbreviations.

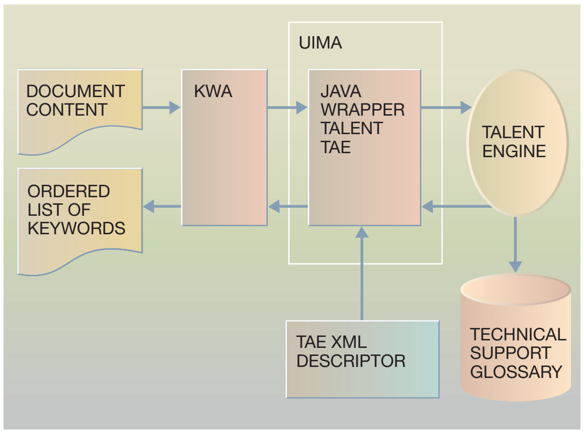
Evaluating the effectiveness of the glossary-extraction process

The glossary we build may not be effective in the search task unless it helps identify *salient* terms (keywords) in the context of interest. In this section we describe the process of extracting salient terms in a document and its implementation in KWA.

The goal of the KWA utility is to find instances of glossary items in the document. Because each item in the glossary is represented as a canonical form and associated variants, the goal is to find not just the instances of the glossary item's canonical form, but the associated variants as well. The approach used to match single-word and multiword canonical forms and their variants in a glossary file (the output of GlossEx) with words in a document is as follows.

- *Inflectional variants match*—KWA matches all inflected forms with lemma forms even though the inflected forms do not exist in the input glossary files.
- *Case-sensitive/insensitive match*—Users can control the case sensitivity in the match. When the "respect_case" option is set to "on" in the configuration, KWA performs an exact case match. Otherwise, it matches all case variations to the target word.
- *Exact syntactic-category match*—KWA matches only the words having the same POS category with the target glossary item. This option, which is always on for reducing many false positives, is especially important for finding instances of abbreviations. For instance, without this option, the abbreviation WAS for WebSphere Application Server would be interpreted as the verb "was" (past tense of "is") and thus would be matched with all variations of the verb 'to be."
- *Biggest span match*—This is an ad hoc procedure for selecting among overlapping spans. When KWA finds that more than one candidate matches with different spans for a glossary item, it selects the biggest span among all candidates.
- *Abbreviation handling*—Technical documents use many abbreviations, so a glossary file contains many abbreviations too. Some abbreviations are ambiguous (IBM is an abbreviation for "International Business Machines," "Intercontinental Ballistic Missile," and "Inclusion Body Myositis").

Figure 6 KWA application schema



The matching rules for abbreviations are as follows. If an abbreviation is unambiguous (i.e., only one definition is found in the glossary file), then KWA always matches the abbreviation with the definition in the glossary file even though the definition does not appear in the text. For instance, when an unambiguous abbreviation, WAS, appears in a document without its definition, KWA returns "Web Application Server" as well as "WAS" as keywords. If, however, an abbreviation is ambiguous (i.e., more than one definition is included in the glossary file) and no definition is found in the document, no disambiguation is done; that is, KWA returns only the abbreviation as a keyword. If the abbreviation is ambiguous and one of the definitions is found in the document, KWA matches the definition with the abbreviation and returns both. If multiple definitions appear in the document, the closest definition from the abbreviation is linked to the abbreviation.

Recently, the IBM Research Division released the Unstructured Information Management Architecture (UIMA) as the middleware architecture for natural language processing (NLP) components and applications.[17,18] KWA is implemented as an application of the UIMA framework and is based on the TALENT TAE.[7] As depicted in Figure 6, KWA processes each input document and returns a sorted list of keywords by instantiating the Java** wrapper TALENT TAE with a TAE XML descriptor that identifies the input requirements, the output specifications, and external resource dependencies, such as dictionaries. For the TALENT TAE, this XML descriptor defines the type of the annotations produced by the TALENT TAE, the

Table 3 KWA results with *all software categories* glossary

	All Manually Selected Keywords (%)	Manually Selected Context-Specific Keywords (%)
Precision	100	50
Recall	10	15
F-measure	18	23

configuration, and the dictionaries and glossaries that the TALENT engine needs to consult. The domain-focused glossary associated with the input document is specified through this XML descriptor and is loaded during the instantiation of the TALENT engine. The result of the analysis of the document by the TALENT TAE contains the annotation objects as specified in the XML descriptor. KWA retrieves the keywords and their variants among the annotated objects, sorts keywords by their scores, and returns the list of the keywords and their variants.

Domain-focused glossaries.

Because a glossary can be viewed as a bag of terms extracted from a collection of documents in a given domain, it naturally depends on the domain. Moreover, not all the terms in a glossary are equally important or salient, as related to the context of a particular document within the domain. For example, the term *computer* is less salient in the *Web Application Development* domain, than the term *WebSphere Studio Application Developer*. We observed that the level of the term's salience in the domain cannot be directly estimated from its statistical attributes in the domain-specific collection of documents. Therefore, manipulating the glossary document base yields limited improvement in the quality of the glossary, as the potential source for salient terms in the given domain.

In this section we introduce the concept of a domain-focused glossary as an enhancement of the plain glossary extracted from a given domain. The process of "focusing" involves emphasizing the weight of context-specific terms within the domain by modifying the attributes of these terms. We describe the details of the glossary-focusing process and show experimental results that substantiate the claim that a domain-focused glossary is an effective source of salient terms for search-related applications, such as the KWA application.

Evaluating the domain-focused glossary. To evaluate the quality of a glossary, we processed selected documents from the Technical Support corpus and compared the keywords automatically extracted by KWA with the keywords manually selected by expert users familiar with the Technical Support corpus. In our experiments, we used documents from one of the *software* categories, chosen for the higher quality of their content.

The experiments we describe below involved three steps. First, using sample documents, expert users manually selected technical terms that seemed relevant for the search task (terms such as *WebSphere Application Server*, *java*, and *computer*). We refer to this set of terms as *all manually selected keywords*. Next, the expert users repeated the same process, but this time they selected only the terms that are representative of the document context. For example, even though the terms *java* and *computer* appear in a document about WebSphere Application Server, they are not selected because they are less important to the context of the document. Contrarily, terms like *WebSphere Application Server* or *APAR PQ78222* are selected by the expert users as salient terms that may represent the document context in the search results, helping users in finding documents relevant to the query. In our experiments we call these salient terms *manually selected context-specific keywords*. At the last step of each experiment, the KWA application is used to automatically select terms from the same sample documents. To get more reliable keywords, we performed keyword filtering based on a selected confidence (C) level threshold. In our experiment the C threshold was set to 2, which allows filtering out most of the "garbage" keywords.

In the first experiment we used the KWA application with a wide glossary extracted from the collection of all software documents (about 158000 glossary entries). Table 3 shows the results of the KWA application performance test based on the manually selected keywords. The first column of Table 3 shows that all keywords automatically selected by the KWA application are included in *all manually selected keywords* (due to the keyword filtering), but the recall and the F-measure are very low. The second column of Table 3 shows that only 50 percent of automatically selected keywords are salient in the context of the documents, and the recall and the F-measure are still low.

Because the results of the first experiment were rather discouraging, we tried to use the KWA application with a narrow glossary extracted from the narrowest category that contains the selected documents (about 2000 glossary entries). The results are shown in Table 4.

Table 4 KWA results with *narrow software category* glossary

	All Manually Selected Keywords (%)	Manually Selected Context-Specific Keywords (%)
Precision	50	30
Recall	30	50
F-measure	38	38

The results show that a wide glossary is not very useful for the KWA application because too few of its high-score entries (with high confidence level) correspond to the given document context. There is an indication that using a narrow glossary can improve the situation, but the observed improvement is not sufficient. To achieve better-quality KWA results, we propose using domain-focused glossaries, which we define in the following way:

1. The domain-focused glossary is extracted from a collection of documents that belong to a reasonably narrow category (domain).
2. Domain specificity scores of the terms that can represent the given domain context are significantly increased, reflecting the level of importance of the terms to the domain; as a result, confidence-level scores for the most important domain-specific terms are also increased.

We use the following formula (equation 5) to calculate the modified domain specificity score (*TD*) and confidence level score (*C*) for the given term *T* (see also equations 2 and 3):

$$TD^*(T) = TD(T) \cdot [K_D(T) + 1],\ K_D(T) = \frac{r}{f(T)^q};$$
$$C^*(T) = a \cdot TD(T) \cdot K_D(T) + C(T). \qquad (5)$$

The proposed modification of the domain specificity score is not applied to all the terms extracted from the given narrow domain collection. To automatically determine which terms can represent the given domain context, we proposed using external domain specific vocabularies, like the IBM Terminology dictionary (see Reference 16) for the given domain. The domain-specificity scores for the glossary terms, which appear in appropriate vocabularies, are increased according to formula (5). The *boosting* coefficient (*K*) in formula (5) is calculated based on the term's frequency (*f*) and two constant parameters (*r, q*), so that context-specific terms that have lower frequency in the given collection would get higher domain specificity. If some important domain-specific term appears only once in a certain document and does not appear in any other documents, the term is likely to be salient for this document. This value of the boosting coefficient *K* provides sufficient increase of the term's domain-specificity score to ensure the term will be selected among top keywords for the given document. We call this mode of the domain-focused glossary a *document view*. We also considered another mode of the domain-focused glossary, where the boosting coefficient *K* is calculated based upon the following formula:

$$K_D(T) = r \times \log_2 [f(T)],$$

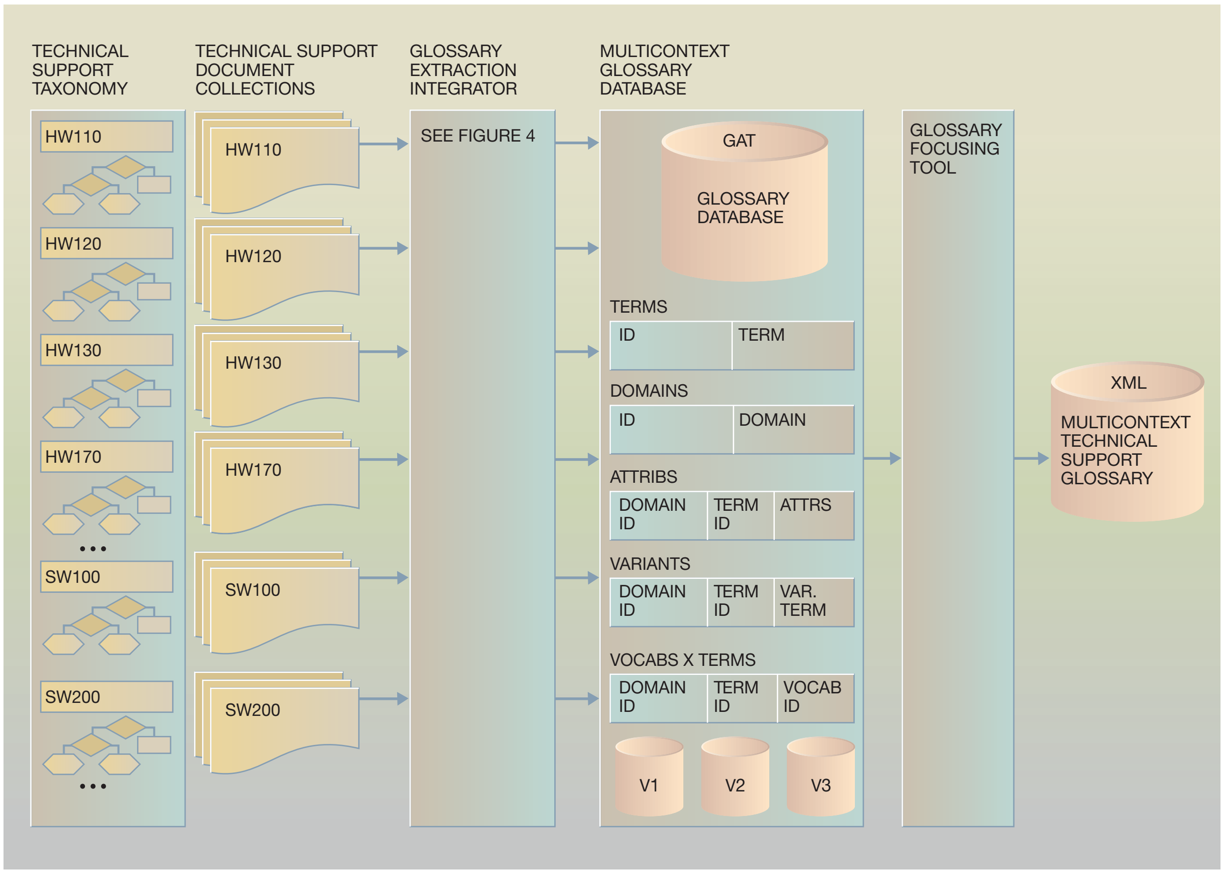
with one constant parameter (*r*), so that context specific terms that have higher frequency in the given collection would get higher domain-specificity scores. This mode may be useful for extracting bags of terms from narrow domain-specific collections of documents. We call this mode of the domain-focused glossary a category view.

The effectiveness of the domain-focused glossary-extraction process depends on the quality of the domain-specific vocabulary. The quality increases as the domain-specific vocabularies are made available to the glossary-extraction process.

Building multicontext glossaries. The process of building domain-focused glossaries introduces new architectural requirements. The final selection of the glossary items requires that selection should be based on the context of the document from which the items are extracted. The same glossary item may appear in different domain-specific glossaries with different confidence levels. Glossary administration tools are essential to manipulate multiple domain-specific glossaries and to establish links to domain-specific vocabulary entries.

In handling the IBM Technical Support glossary we modified the existing Glossary Administration Tool (GAT) to support the multicontext glossary manipulations described above. All IBM Technical Support documents are associated with IBM products, and IBM

Figure 7 Building a multicontext glossary



adopts a categorization scheme called the Technical Support taxonomy. At creation time, each Technical Support document is assigned to certain nodes of the product taxonomy. Figure 7 shows the process of building multicontext glossaries from Technical Support document collections. The glossary extraction tool, as described in the section "Overview of glossary extraction process," is used to generate the glossary items from the collection of documents belonging to each category. All category-specific glossary-candidate files are loaded into the multicontext glossary database, which includes the information of the document categories, the total number of documents and terms in each category, the statistical information (frequency, confidence level, number of words in term, etc.) of each domain's term and its variants (inflection, misspelling, abbreviation, etc.). The glossary database schema includes one table for all the terms and a table for domains (categories), as shown in Figure 7. Because each term may have different sets of statistical attributes in different domain-specific collections, so the database allows storing one set of attributes for each term in each domain in the Attributes table. In the same way, the database allows the storing of several domain-specific sets of variants for each term in the Variants table. Eventually, the multicontext glossary database creates a cross-reference table that contains information about domain-specific terms that appear in one of appropriate domain-specific vocabularies: V1, V2, V3, and so forth.

The domain-focused glossary, as described in the section "Domain-focused glossaries," is created first, by marking terms that belong to domain-specific dictionaries or vocabularies (shown as V1, V2, V3, etc. in Figure 7), second, by storing this information in the database cross-table of vocabularies and terms, and, finally, by recalculating the domain specificity (TD) and the confidence level score (C) of the marked terms by using equation (5) in the subsection "Evaluating the domain-focused glossary."

Future work. One possible application of the domain-focused glossary is the improvement of document relevancy ranking in corporate search systems. Traditional full text search systems use the TF-IDF formula[19] for calculating document relevancy scores/ranking. According to this formula, the document relevancy score S, related to the search term T, is calculated as

$$S(T) = F(T) \times \log_2[N/n(T)],$$

where $F(T)$ is the frequency of term T in the document, N is the number of documents in the collection, and $n(T)$ is the number of documents where term T appears at least once.

This formula works well for general web-based search systems, but it may not work for corporate search systems, which use strong classification of documents based on proprietary corporate taxonomies. The following example illustrates some deficiencies of this method in a corporate technical-support search system.

Example: consider a search system with 500 000 documents indexed, where

1. 1000 documents contain term *T2*—"WAS" (WebSphere Application Server);
2. 10 documents among them contain term *T1*—"NoResourceException."

Assume that there is one document (A) that contains two occurrences of term *T1* and 10 occurrences of term *T2*, and there is one document (B) that contains one occurrence of term *T1* and one occurrence of term *T2*.

A user submits the following query: *NoResourceException in WAS*. The system calculates scores of documents A and B for terms *T1* and *T2*, based on the TF-IDF formula, as follows:

$$\text{Score\_A}(T1) = 2 \times \log_2(500\,000/10) = 30.2;$$

$$\text{Score\_A}(T2) = 10 \times \log_2(50\,0000/1000) = 89.7;$$

$$\text{Score\_B}(T1) = 1 \times \log_2(50\,0000/10) = 15.6;$$

$$\text{Score\_B}(T2) = 1 \times \log_2(50\,0000/1000) = 9.$$

According to these scores, the search system puts document A at the top of the hitlist, and document B goes to the bottom. Now, let us assume that document A contains a customer problem report, and document B contains a link to the patch that should be applied to resolve the problem reported by the customer. In a traditional search system the user, most likely, will never open document B because it does not appear at the top of the hitlist, or even on the first page of the hitlist.

The proposed approach changes the way the search system calculates document relevancy scores by introducing context-dependent weights of search terms. The scores for given search terms are calculated based on weights assigned to these terms in accordance with their salience in the given context. If in the above mentioned example the document scores were calculated based on weights assigned to terms *T1* and *T2* in the context of *WAS runtime exceptions*, then both documents A and B would have similar scores, because $WC(T1) \gg WC(T2)$, where $WC(T)$ is the weight of the term T in the given context C.

To assign appropriate weights to query terms the proposed method uses domain-focused glossaries that are created based upon the corpus of documents, using available corporate taxonomies and specialized technical vocabularies. As shown in the section "Domain-focused glossaries," in a domain-focused glossary each term's score is calculated based on the term's salience in the given context. For example, the term *NoResourceException* has a high confidence level in the context of *WAS runtime exceptions*, but may be ignored in the context of *WAS product sales*, whereas the term *WAS* is more salient in the latter context.

Conclusion

In this paper, we discussed the need for enhancing traditional glossary extraction processes for IBM's technical support corpus, the methods we used, and the results we achieved. Doing so increased the usefulness of the search and delivery system for IBM's technical information as well as customer satisfaction with the IBM support site. This also improved the tools we use for managing multidomain glossaries.

Central to this work, we introduced the idea of domain-focused glossaries. We discussed the importance of tuning the process of generating glossaries to the specificity of the corpora. We introduced the method of biasing or focusing a glossary to the context of the domain from which the glossary is extracted. This process requires building dictionaries of domain-specific terms. The method of biasing comprises modifying the weights of the glossary terms consistent with the domain context information.

In our prototype, we found that using domain-focused glossaries in the KWA application may yield improved document-relevancy ranking by appending salient terms to the document meta-data. We also considered improving the relevancy ranking algorithm based upon the weights of the salient terms in the document and presented this idea as future work. We explored improving customer-search goal attainment by employing keyword analysis based on focused glossaries. This is particularly important for the IBM Technical Support information search and delivery system, where documents belong to many different hardware and software categories.

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Aventail Corporation, Borland Software Corporation, or Sun Microsystems, Inc.

Cited references

1. Text REtrieval Conference (TREC), National Institute of Standards & Technology, U.S. Department of Commerce, http://trec.nist.gov.
2. Y. Park, R. J. Byrd, and B. K. Boguraev, "Automatic Glossary Extraction: Beyond Terminology Identification," *Proceedings of the Nineteenth International Conference on Computational Linguistics* (2002), pp. 772–778.
3. *IBM Support and Downloads*, IBM Corporation, http://www.ibm.com/support.
4. Y. Doganata, Y. Drissi, T. Fin, G. Brown, M. Kim, and L. Kozakov, "dBlue — An Advanced Enterprise Information Search and Delivery System," *WebSphere Developers Journal* **1,** No. 9, 6–10 (October 2002).
5. B. K. Boguraev and M. S. Neff, "Lexical Cohesion, Discourse Segmentation and Document Summarization," *Proceedings of the 6th Conference on Content-Based Multimedia Information Access (RIAO'2000)*, Center for the Advanced Study of Information Systems, Inc. (2000).
6. C. Wolf, S. Alpert, L. Kozakov, J. Vergo, Y. Doganata, and C-M. Karat, Comparing Search Results Summaries for Technical Support Documents: A User Study, *IBM Research Report*, RC22719, IBM Corporation, Yorktown Heights, N.Y. (2003).
7. *Text Analysis and Language Engineering*, IBM T.J. Watson Research Center, IBM Corporation (2001), http://www.research.ibm.com/talent/.
8. *TIPSTER/SUMMAC Summarization Evaluation*, T. Hand and B. Sundheim, Editors, *Proceedings of the TIPSTER Text Phase III 18-Months Workshop*, Fairfax, VA, 1998, Defense Advanced Research Project Agency (1998).
9. I. Mani, T. Firmin, D. House, G. Klein, B. Sundheim, and L. Hirschman, "The TIPSTER SUMMAC Text Summarization Evaluation," *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics* (1999), pp. 77–158.
10. J. Justeson and S. Katz, "Technical Terminology: Some Linguistic Properties and an Algorithm for Identification in Text," *Natural Language Engineering* **1,** 9–27 (1995).
11. L. Karttunen, J. Chanod, G. Grenfenstette, and A. Schiller, "Regular Expressions for Language Engineering," *Natural Language Engineering* **2,** No. 4, 305–328 (1996).
12. B. Boguraev, "Towards Finite-State Analysis of Lexical Cohesion," *Proceedings of the 3rd International Conference on Finite-State Methods for Natural Language Processing*, INTEX-3, Liege, Belgium (2000).
13. Y. Park and R. Byrd, "Hybrid Text Mining for Matching Abbreviations and their Definitions," *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*, Intelligent Information Systems Institute, Cornell University (2001), pp. 126–133.
14. Y. Park, "Identification of Probable Real Words: An Entropy-based Approach," *Proceedings of an ACL Workshop on Unsupervised Lexical Acquisition*, Association for Computational Linguistics (2002), pp. 1–8.
15. L. Dice, "Measures of the Amount of Ecologic Associations between Species," *Journal of Ecology* **26,** 297–302 (1945).
16. IBM Terminology, IBM Corporation (2001), http://www.ibm.com/ibm/terminology/.
17. A. Spector, "Architecting Knowledge Middleware," *Proceedings of the Eleventh International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, May 2002, ACM, New York (2002), http://www2002.org/spector.pdf.
18. D. Ferrucci and A. Lally, "Accelerating Corporate Research in the Development, Application and Deployment of Human Language Technologies," *Proceedings of the HLT-NAACL 2003 Workshop on Software Engineering and Architectures for Language Technology Systems*, Edmonton, Canada (2003), pp. 68–75.
19. R. Baeza-Yates and B. Ribiero-Neto, *Modern Information Retrieval*, Addison-Wesley, Reading, MA (1999).

Accepted for publication February 28, 2004.

Lev Kozakov *Research Division, IBM Thomas J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598 (kozakov@us.ibm.com).* Dr. Kozakov is a research staff member at the Watson Research Center. He received a Ph.D. degree in applied mathematics and computer sciences from M.V. Lomonosov Moscow State University in 1983. He has worked in many areas, including dynamic systems, applied statistics, information management systems, man-machine interface, and object-oriented design and programming. His current research interests include information-management frameworks and natural-language-processing technologies. He has published in various fields of computer science and applied mathematics and holds several patents.

Youngja Park *Research Division, IBM Thomas J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598 (young_park@us.ibm.com).* Dr. Youngja Park received a Ph.D. degree in computer science from Yonsei University in Seoul, Korea. Her research interests lie in natural language processing, and in particular, in unsupervised lexical knowledge acquisition, computational terminology, information extraction, information retrieval, and the semantic Web. She is also interested in genetic algorithms, and especially in applying genetic algorithms to clustering problems.

Tong-Haing Fin *Research Division, IBM Thomas J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598 (thfin@us.ibm.com).* Dr. Fin is a senior software engineer at the Watson Research Center. He received a Ph.D. degree in computer science from Tokyo Institute of Technology, Tokyo, Japan, in 1982. He has worked in many areas, including computer-supported cooperative work, real-time conferencing and meeting systems, and team-collaboration applications. He is the author of a number of publications in computer science and holds several patents. His current research interests include natural-language-processing technologies, team-collaboration technologies, and designing and prototyping innovative applications in the area of unstructured information management.

Youssef Drissi *Research Division, IBM Thomas J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598 (youssefd@us.ibm.com).* Mr. Drissi is an advisory software engineer at the Watson Research Center. He received an M.S. (diplome des arts et manufactures) degree in computer science from Ecole Centrale de Paris, France. He joined the Watson Research Center in 1998 and worked in various areas, including text analytics, knowledge management, search systems, electronic services, predictive analysis, meta-learning, and Internet messaging frameworks. He is the author of several publications and patents. His current work involves designing and prototyping innovative solutions, applications, tools, and utilities in the area of unstructured information management.

Yurdaer Doganata *Research Division, IBM Thomas J. Watson Research Center, PO Box 704, Yorktown Heights, New York 10598 (yurdaer@us.ibm.com).* Dr. Doganata is the manager of the Information Management Solutions group at the Watson Research Center in Hawthorne, New York. He received B.S. and M.S. degrees from the Middle East Technical University, Ankara, Turkey, and a Ph.D. degree from the California Institute of Technology, Pasadena, California, all in electrical engineering. He joined the Watson Research Center as a research staff member in 1989 and worked on projects in many diverse areas, including high-speed switching systems, multimedia servers, intelligent transportation systems, multimedia collaborative applications, e-services, and information search and retrieval systems for technical support. His current work involves designing and prototyping innovative solutions, applications, tools, and utilities in the area of unstructured information management.

Thomas A. Cofino *Research Division, IBM Thomas J. Watson Research Center, PO Box 704, Yorktown Heights New York 10598 (cofino@us.ibm.com).* Mr. Cofino is a research staff member and Senior Manager at the Watson Research Center. He joined the company in 1969 and the Research Center in 1974. Mr. Cofino has held numerous technical leadership and management positions within IBM, including the Research Division. He led projects on RFID systems, User Experience and User-Centered Design, text analytics and search systems, and user-interface management systems, including the Guest Services System at Expo 92 in Seville, Spain. He is the author of a number of publications and holds 14 patents. His current work involves designing and prototyping innovative information-retrieval applications in the area of unstructured information management. He holds an M.B.A. degree in management information systems from Iona College, New Rochelle, New York.