RC23234 (W0406-038) June 8, 2004
Computer Science

IBM Research Report

Enabling Text Processing Applications with Automatic Glossary Extraction

Youngja Park, Roy J. Byrd, Branimir K. Boguraev
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Enabling Text Processing Applications With Automatic Glossary Extraction

Youngja Park, Roy J. Byrd and Branimir K. Boguraev
IBM Thomas T. J. Watson Research Center
P.O. Box 704, Yorktown Heights, NY 10598, USA
{young_park, roybyrd, bran}@us.ibm.com

Glossary items in specialized technical genres exhibit many of the problems associated with multiword expressions. This paper describes recent work on automatic glossary identification and extraction, and examines a number of issues arising when a glossary extraction technology is used as an enabling capability for a broad range of text processing applications. We outline an algorithm for glossary item identification, and discuss how a particular encapsulation of it addresses architectural, representational, and interface issues emerging during harnessing the technology for a number of applied tasks.

1. Introduction

Nominal multiword expressions exemplify a particularly productive phenomenon in natural language, with manifestations in numerous text genres. Of special interest to our research is the class of multiword expressions constituting "glossary items". Pervasive in technical prose, glossary items carry a substantial fraction of the description of a domain of interest (both informal, e.g., in the shape of glossaries and index lists, and formal, such as ontological representations).

It is unreasonable to expect that an application in such a domain could be informed by a readily precompiled list of glossary items. Domain-specific glossaries need to be dynamically acquired: ideally, from a background document collection representative of the domain. Previously, we have reported on a method for the identification and extraction of glossaries from such document collections (Park, Byrd, & Boguraev, 2002). In this paper, we expand upon that work; additionally, we look at what it takes to bridge the gap from developing the underlying technology of glossary extraction to making that technology transparently usable by a broad range of applications.

Examples of such applications include information extraction in technical domains, document indexing and search, (possibly in interaction with clustering and cluster labeling), back-of-the-book indexing, federation of heterogeneous document collections (possibly complemented by keyword extraction and summarization), machine translation (possibly in tandem with others from this list), and automated construction of domain taxonomies and ontologies (for use by, e.g., higher-level reasoning components).

We look at the problem of glossary identification and extraction from the perspective of configuring such applications in ways which optimally use, and reuse, the technology. Some of the challenges here are architectural, others are representational, yet others emerge at the interface where glossaries identified are to be made available to other system functions.

For instance, part-of-speech tagging and (shallow) parsing are essential components of our glossary extraction algorithm (Park, Byrd, & Boguraev, 2002); at the same time, applications typically need to deploy some tagging/parsing capability *after* glossary identification, using knowledge of glossary items as regular syntactic units. Thus, the ability to configure a parser to abstract over units of different granularity, and to make a glossary extraction component co-resident with such a parser and capable of communicating the full complement of its analyses to it, is crucial for enabling a larger application.

Similarly, abbreviation processing can be viewed as an equally important part of glossary identification, while remaining a separate, identifiable, function of semantic category labeling.

To make interactions like these possible, we need to consider both the architectural ramifications for a text processing framework, and some underlying features of the representation, not only of identified glossary items, but in general of linguistic entities flowing through the system. Frequently, an application would need to use a glossary list derived not from a background collection, but constructed 'on demand', from the very documents being processed; this means that processes of glossary extraction and glossary use are not as cleanly separable as typically assumed, and that inter-operability between the glossary extraction technology and other application components is essential.

We introduce a distinction between *discovery* and *recovery* modes of system operation—distinguishing between, broadly speaking, dynamic lexicon acquisition and autonomous use of it as a resource—and define a design where the two can co-exist in any particular system configuration. Situating a glossary extraction function in a larger text processing system so it meets the challenges listed above is made easy by particular features of a text processing environment. We outline some of these in the next section.

2. Text Processing Infrastructure

An example of a system whose architectural features facilitate both the encapsulation and subsequent use (and reuse) of a glossary extraction capability can be characterized as a componentized, pipelined processor, with individual modules (annotators) organized in a reconfigurable pipeline (Neff, Byrd, & Boguraev, 2004). Annotators communicate only indirectly, by means of annotations posted to, and read from, a common analysis structure.

2.1. Annotations-Based System Architecture

Annotations, which are essentially compliant with the broadly established notion developed in (Bird & Liberman, 2001), encapsulate data with respect to an overall data model; for us, the literal notion of annotation as a span of text is complemented by a lexical cache, a vocabulary, and other shared resources. Particularly relevant to this discussion are the principles of *common representational substrate* (which promotes uniform API's for, e.g., both posting results during discovery, and reading from resources during recovery), *independence of individual annotators' processing* (which facilitates retargeting over input streams with different granularity, and reusability of annotators), and *separation of data and resources from function* (which allows, e.g., reconfiguring of general algorithms for different data).

In such an architecture, applications are realized as specific instances of pipelines, incorporating sequences of linguistic analysis functions appropriate to any particular application. As Neff, Byrd, & Boguraev (2004) argue, this allows us to achieve several crucial goals: through reconfiguration, we can address the needs of diverse applications; by using shallow and finite-state analysis (see below), we obtain the speed and scale required by realistic applications; with analysis methods which are either trainable (e.g., HMM part-of-speech taggers) or easily parameterized (e.g., finite-state transducers), we can customize our analyzers and address applications in multiple languages.

Such architectural and representational features make it possible to provide—on an infrastructure-wide basis—glossary extraction services as an encapsulatable technology, usable by diverse applications with, or without, reconfiguration. Thus, for instance, the glossary extraction algorithm (described in Section 3 below) is packaged as an annotator; it naturally incorporates earlier analyses by, e.g., lexical lookup and part-of-speech tagging; it operates over (internally identified) candidate glossary items represented as annotations; and it publishes its discovery results (canonical form, variants, morphosyntactic make-up, and so forth) to the vocabulary store. Once in the vocabulary, the output of glossary extraction is available for recovery by (any) subsequent annotator(s) in the particular application pipeline; naturally, recovery can be as simple as just streaming the glossary list to a persistent database.

Thus, by storing the output of the glossary annotator in the vocabulary store, and making that available as a resource, applications can use a glossary list in a variety of ways. They can, for example, abstract away from annotation features, and directly query properties of glossary items; they can access glossary items as if they were annotations, via an annotations API; or they can even re-introduce glossary items as annotations in the input stream. This meets the needs of efficient, streamlined access with a natural way of making the output of glossary extraction transparently available to higher level, downstream, linguistic annotators such as a parser (see Section 5 below).

The glossary extraction algorithm itself is, as we shall see, decomposable, separating the identification of candidate items from determination of true glossary items based on collection of properties. To the extent that glossary items do reflect specifics and idiosyncracies of different domains, glossary extraction needs to be sensitive to these. In the framework we adopt, and in particular given our strategy for using finite-state technology for the identification, it is straightforward to enable adaptation of the basic approach and algorithm to different domains and text genres, simply by swapping grammars for candidate glossary items. In fact, the very task of glossary identification can be slightly modified—by means of grammar swapping alone—to derive related functions of, e.g., technical terminology identification or compilation of an initial list of candidate entries for a back-of-the-book index as variations of the basic algorithm, operating on data sets with different properties, better suited to the task at hand. We discuss such adaptation strategies in Section 5.

2.2. Annotations-Based Finite-State Subsystem

One essential infrastructure component which facilitates our approach is a finite-state (FS) transduction capability. There are a number of reasons for using FS technology for a pattern identification task like ours: see, for instance, (Karttunen *et al.*, 1996). We are particularly concerned with issues of reusability, adaptability and portability of grammars, as essential characteristics of rapid application configuration.

For the task of glossary extraction in particular, *reusability* means being able to leverage a general noun phrase grammar, with known properties and coverage, for the purpose of identifying nominal glossary candidate expressions; reusability also means parameterizing such a grammar for—and thus making it aware of—domain- and/or, genre-idiosyncratic expressions, which can then be naturally incorporated into larger syntactic processing cascades. *Adaptability* speaks to concerns related to different domains and/or applications (where the overall contour of a glossary item would remain substantially unchanged, but for relatively minor/local domain-specific modifications). *Portability* largely concerns retargeting grammars for different languages. This last consideration is essential for a number of applications, where glossary extraction needs to be carried out over multi-lingual text collections. In principle, it is easier to adapt an implementation to a different language, if phrasal patterns are specified as linguistic abstractions, interpreted by a language-independent engine. Moreover, using FS techniques, it is possible to focus on a class of relatively simple (and, specifically, context-independent) linguistic patterns such as noun phrases, verb groups, subordinate clauses and specify these as cascaded constraints, largely language-independent for compiling language-independent “chunkers” (Kinyon, 2001).

A pipeline architecture with an annotations-based representation model requires a special-purpose FS parsing regime. There is no *a priori* limit on the annotations’ number and types (nor on their properties and attribute values within): words, named entities, special tokens, domain-specific terms, phrases, chunks, sentences, and so forth are all annotations. While enabling the sequencing, and integration, of numerous annotations, and allowing for very tight coupling among arbitrary sets of linguistic analysis components, this representation effectively “hides” the view of a document as a sequence of like items (typically characters or tokens), replacing it with a lattice of diverse annotation types spanning over arbitrary, possibly discontiguous, possibly overlapping, text fragments. This raises certain issues—operational and notational—concerning FS-based matching *over the underlying annotations lattice*.

Boguraev (2004) describes a generalization of the notion of character-based finite-state transduction, which supports writing patterns in terms of annotations, and which implements a finite-state executor configured to reading from, and writing to, an annotations store. This approach substantially improves the efficiency of an FS-based recognizer, a critical consideration for any application. More importantly, however, it makes it very natural to implement finite-state cascades, both for identification of glossary items (such as the one employed by our glossary extraction procedure, incorporating look-up, named entity identification, part-of-speech tagging, and candidate glossary item identification; see Section 3), and for their subsequent use by autonomous applications (such as, for instance, deriving candidate index entries for a back-of-the-book indexer by means of a higher level FS cascade; see Section 5).

3. Glossary Extraction Algorithm

The goal of glossary extraction is finding words or phrases that name and describe domain concepts in documents for a particular domain. Many attempts have been carried out to automatically identify units of terminology or multi-word expression. One approach is to use a regular expression or grammar specifying the syntactic structure of the target terms (Justeson & Katz, 1995). The other approach is to find cohesive word n-grams, with or without part-of-speech information, based on statistical information (Church & Hanks, 1990; Dunning, 1993; Smadja, 1993; Dias *et al.*, 2000).

The identification of candidate glossary items, however, poses challenges which cannot be met by a simple lexical filter or n-gram approach. For one thing, the noun phrase structure for glossary items is more complex than that of a term phrase introduced in (Justeson & Katz, 1995). A bigger challenge is that of extracting "domain-specific" and "concise" terms in the given documents. In this section, we describe a hybrid method, which applies both linguistic and statistical knowledge, for extracting glossary items to address these issues and thus to improve the performance of various text processing applications.

The overall outline of the algorithm is as follows; First, the algorithm identifies candidate glossary items by using syntactic grammars and also a set of entity recognizers. To extract more cohesive and domain-specific glossary items, we conduct prenominal modifier filtering and various glossary item normalization. Finally, the glossary items are ranked based on their domain-specificity and cohesion.

3.1. Identification of Candidate Glossary Items

We apply a set of nominal expression grammars with the FST machine described in section 2.2 over a POS-tagged corpus. The nominal expressions we aim to recognize as glossary items consist of zero or more prenominal modifiers such as adjectives, verb participles or nouns and a headword of a noun (e.g., "*a multi-word expression*") or a cardinal (e.g., "*Windows 2000*"). Therefore, glossary items can be regarded as a subset of general noun phrases (NP's), which are the domain of NP chunking or shallow parsing tasks. For instance, pronouns and noun phrases with prepositions are not considered as glossary items. Noun phrases consisting only of pronouns (e.g., "*it*", "*they*" and "*which*") have little value as domain terminology, and thus discarded. Whereas noun phrases with prepositional phrases are broken into their base noun phrases, and the base noun phrases are considered as candidate glossary items. For example, we extract two candidate glossary items, "*structural parts*" and "*Boeing's 747 jetliners*" from "*structural parts for Boeing's 747 jetliners*".

A challenge for the nominal expression recognition task is developing a set of grammars which provide both accuracy and coverage. A liberal grammar may produce many false noun phrases, while a conservative grammar may miss many instances of valid noun phrases. To address this trade-off, we developed a set of context-sensitive grammars to identify noun phrases. When a nominal expression has a strong cue for being a noun phrase in the context, we apply more liberal grammars. For cases where there is no cue, the grammars are more conservative to extract accurate noun phrases.

For instance, as shown in the examples (1)–(3) below, verb participle forms can be used as premodifiers. However, verb participle forms are frequently used as finite verbs in sentences like example (4). We examine the context to decide if a verb participle form is used as a premodifier or a verb. When a verb participle follows a determiner or a verb, then the verb participle is likely a premodifier. Without any cue, a premodifier beginning with a verbal participle is considered less likely to be part of the nominal expression.

(1) The troops are rebuilding *ruined military sites*.
(2) The *ruined military sites* were rebuilt.
(3) Iraq's *ruined military sites*.
(4) The troop ruined *military sites*.

Table 1 shows some examples of candidate glossary items recognized with the nominal expression grammars from documents in the automobile, finance, and intelligence domains.

Table 1
Examples of candidate glossary items from multiple domains. A denotes premodifiers (adjectives and verb participles), R stands for adverbs, N denotes nouns, and C denotes conjunctions.

Structure	**Terms**
NN	Coordinating Committee
ANN	rear wiper blade
NNN	emission control system
RAN	highly enriched uranium
NNN	http://www.pc.ibm.com/support Web site
AANN	other qualified service technician
AANN	hybrid nuclear/diesel-electric propulsion unit
AANN	national over-the-counter trading yesterday
ACAAN	ambient and wide open throttle
ANNNN	Panamanian leader Manuel Antonio Noriega
NNNNN	Shearson Lehman Treasury index 1300.16
AACAN	certain frontal or near-frontal collision
ACANNN	Grim and tight-lipped , Sen. Robert Dole
ANNNNN	composite New York Stock Exchange trading
NNNNNN	Guangdong Nuclear Power Joint Venture Company
RRAANN	most actively traded 30-year Treasury bond
ANCANN	biggest highest-grade and lowest-cost uranium mine
AAAAAN	double B-minus convertible exchangeable preferred stock

The advantage of FST-based glossary identification is in its flexibility and adaptability. The main glossary extraction system is very general, and no domain-specific or language-specific knowledge is incorporated. When the domain or the application changes, the syntactic structure and/or the nature of glossary items of interest may be different from other domains or applications. Since the grammar is an externally provided resource, our system can replace an existing grammar with another without further system modifications or adjustments.

In addition to the FST-based recognizers, we also use recognizer for out-of-vocabulary (OOV) words and for abbreviations and their full forms. The proper recognition and treatment of these terms is important to applications because such terms are usually domain-specific, and also because multi-word terms with abbreviations usually indicate that the terms are known multi-word expressions. We developed an abbreviation pattern-based algorithm to match abbreviations and their definitions in the text (see, (Park & Byrd, 2001) for the full description). When a multi-word term has a abbreviation, the term is included in the candidate glossary set. Furthermore, the full form is considered as a multi-word unit and thus is exempt from the premodifier filtering step introduced in section 3.3.1. We also pay attention to the fact that some OOV words are new domain-specific words, while many other OOV words are proper names or arbitrary strings. We decide if an OOV word is a real word or not on the basis of its morphological constituents and the entropy of the character sequences in the word (Park, 2002). If an OOV word is regarded as a real word, it is also considered to be a domain word. Thus, the word is retained as a candidate glossary item and also receives a high domain-specificity value (see Section 3.4).

3.2. Entity Recognition for Glossary Extraction

Documents contain many nominal expressions denoting named entities such as person, location and company names and also domain objects such as product names, computer commands or addresses. These entities are not very meaningful as domain-specific terminology and also can deteriorate the quality of other nominal expressions. It is not uncommon for noun phrase recognition systems, both carefully written grammar-based (Justeson & Katz, 1995; Frantzi & Ananiadou, 1999; Park, Byrd, & Boguraev, 2002) and machine learning-based (Ramshaw & Marcus, 1995), to err by identifying two different noun phrases occurring in sequence as a single noun phrase. Among candidate glossary items shown in Table 1, "*national over-the-counter trading yesterday*", "*Panamanian leader Manuel Antonio Noriega*"and "*http://www.pc.ibm.com/support Web site*" are examples of this case. These terms are less cohesive and may be semantically incoherent.

We argue that by filtering out "not-useful" terms and by separating named or domain entities from containing noun phrases, we can obtain a better set of glossary items. Our glossary extraction system supports external entity recognizers (in the form of both grammars and programs) for identifying particular entities of interest according to the applications' purposes. Those entity recognizers and the glossary extraction system can run in a cascaded way. Entity recognizers post annotations of their target entities over the documgent, and then the glossary extraction system respects these annotations so that they are not included as glossary items or as a part of larger glossary items.

In this work, we discard the following forms from the set of candidate glossary items: person names, place names, time expressions, special tokens such as URL's and computer path names, and noun phrases having more than 6 words (pre-determiners and determiners are not included).

With entity recognizers for person names, time expressions and URLs, the system recognizes six nominal expressions for the three examples above: "*national over-the-counter trading*", "*yesterday*", "*Panamanian leader*", "*Manuel Antonio Noriega*", "*http://www.pc.ibm.com/support*", and "*Web site*", and take only "*national over-the-counter trading*", "*Panamanian leader*", "*Web site*" as candidate glossary items.

3.3. Normalization of Glossary Items for Domain-Specificity and Cohesiveness

The normalization strategies described in this section aim to further increase the cohesiveness of glossary items and to better distinguish domain-specific glossary items from generic terms. To this end, we propose automatic premodifier filtering and aggregations of various glossary items. Our insight is that prenominal, especially generic, modifiers diminish the cohesiveness of a glossary item. On the other hand, glossary items which are represented by many different forms, such as abbreviations, tend to be more cohesive and domain-specific.

3.3.1. Premodifier Filtering

Many premodifiers in noun phrases, even in domain-specific noun phrases, act as general-purpose modifiers instead of conveying domain-specific information. There are two problems in including all premodifiers in glossary items. First, these adjectives may weaken the domain-specificity of the term they modify. As a result, the terms may have lower confidence values. Second, there may exist many essentially identical forms with slightly different modifiers, which we don't want to keep separately in a domain-specific dictionary. For example, we would like to have a term "*vehicle*" instead of having three different terms such as "*particular vehicle*", "*other vehicle*" and "*real vehicle*". Thus, if a candidate noun phrase contains a generic premodifier, we remove the modifier from the noun phrase and take the remaining noun phrase as a candidate glossary item.

How can we decide if a premodifier is domain-specific or generic? The easiest way might be to keep a list of generic premodifiers and to remove them from candidate forms. However, some premodifiers may be domain-specific in one domain but not in others. When the domain changes, the premodifier list would need to be changed as well. In this work, we automatically decide whether a premodifier should be filtered based on the domain-specificity of the premodifier and the association of the premodifier and the noun it modifies (the first noun following the premodifier).

Domain-specificity means how much a word represents information about the domain and is computed by the relative probability of the occurrence of the word in a domain-specific document and in a general corpus (section 3.4). The association of a premodifier (a) and a noun (n) is calculated by the conditional probability of the occurrence of the headnoun given the premodifier, $p(n|a)$. If a premodifier has high domain-specificity and high association with the headnoun, the modifier remains as a part of the glossary item.

Upon application of the filtering process, the premodifiers *other, certain* and *most* are removed from noun phrases "*other qualified service technician*", "*certain frontal or near-frontal collision*" and "*most actively traded 30-year Treasury*" respectively (examples from Table 1).

3.3.2. Glossary Item Aggregation

A glossary item may appear in different ways in text—for instance as abbreviations or with alternate spellings. The recognition and aggregation of these different forms provide applications with a consistent set of glossary items. Also, glossary items with different representations tend to be more domain-specific. We also combine the frequencies of all different forms, so that terms having many occurrences of many variant forms may have a higher confidence value. The variant forms we take into account are abbreviations, misspellings, orthographic variations, and inflectional variations.

Abbreviations. Abbreviations are especially meaningful in domain documents, and signify that their full forms are very cohesive, i.e., highly likely multi-word expressions. Our earlier work (Park & Byrd, 2001) describes how to identify abbreviations and their full forms in detail. This algorithm can learn the abbreviation patterns commonly used in a certain domain and apply the patterns for matching abbreviations and full forms. Some abbreviations and their definitions found in domain documents are:

- 4H — *four-wheel drive high,*
- ATVL — *Advanced TV-Video Laboratories,*
- F&M — *Facilities and Materials,*
- DOHC — *Double Overhead Cam.*

Misspellings and Alternative Spellings. We use the string edit distance mechanism (Levenshtein, 1962) and the *spell-aid* function in IBM Dictionary and Linguistic Tools (IBM, 2001), which returns up-to 6 possible correct words for a misspelling. For a OOV word, we run the *spell-aid* function to obtain possible correct forms. If a candidate correct form and the misspelled word has the edit distance less than 2 and the candidate correct form appears in the document, we treat them as spelling variations. For example, "*anaesthesia*" and "*anasthesia*" are found as spelling variants of "*anesthesia*".

Orthographic Variants. This variation links two glossary items which consist of the same words but have different separators, such as spaces, hyphens or dashes as shown below:

- *audio/visual input — audio-visual input,*
- *electro-magnetic clutch — electromagnetic clutch,*
- *Mass Airflow Sensor — Mass Air Flow Sensor.*

Inflectional Variants. Finding inflectional variations for multi-word terms with different orthographic features and misspellings is not straightforward. We first sort candidate glossary items in alphabetical order. Then, for each candidate glossary item, we examine a number of neighbors and apply morphological rules to find if the two glossary items are inflectional variations.

- *Fog lamp — fog lamps, foglamps,*
- *CD ROM — CD ROMs, CD-ROMs,*
- *multiprocessing ps/2 — multiprocessing ps/2s.*

3.4. Glossary Item Ranking and Selection

As the final step, we score each glossary item based on how much an item is related to the given domain (*domain-specificity*), and the degree of association of all words in the glossary item (*cohesion*). The confidence value of a glossary item, $C(T)$, is defined as a linear combination of its domain specificity, $D(T)$, and cohesion, $A(T)$.

$$C(T) = \alpha \cdot D(T) + (1 - \alpha) \cdot A(T) \tag{1}$$

where α is a constant which controls the relative contributions of $D(T)$ and $A(T)$ respectively.

Domain Specificity. If an item is used more often in a domain-specific document than in a non-domain document collection, it is likely a domain-specific term (Damerau, 1993). We define the domain-specificity of a glossary item T, $D(T)$, as the average of the domain-specificity of all the words in the glossary item.

$$D(T) \;=\; \frac{\sum_{w_i \in T} \log \frac{P_d(w_i)}{P_c(w_i)}}{\mid T \mid} \tag{2}$$

where $\mid T \mid$ is the number of words in glossary item T, $p_d(w_i)$ is the probability of word w_i in a domain-specific document, and $p_c(w_i)$ is the probability of word w_i in a general document collection.

Note that $p_c(w)$ can be 0 if w is unseen in the general corpus. If w is determined to be a domain word by the OOV recognition algorithm (Park, 2002), the frequency of the word is set to the lowest frequency of words in the corpus so that the word can have a high domain specificity. On the other hand, if the word is not a real word, the frequency of w is set to the highest frequency resulting in a low domain-specificity.

Cohesion. The cohesion of a glossary item denotes how likely the component words in the item appear together as one unit. Our insights for cohesive glossary items are as follows. First, the component words in a cohesive glossary item tend to appear together often, as noted in (Church & Hanks, 1990; Dunning, 1993). Second, more frequent glossary items tend to be more domain-specific; that is, if two glossary items have the same relative frequency as a whole and individually, the glossary item with a higher frequency is regarded as more cohesive. Third, the relative cohesion values of different glossary items should be able to be compared regardless of the length of glossary items. For example, the measure can compare the cohesion of a two-word term with that of a four-word term.

We generalize the Dice coefficient (Dice, 1945) in such a way that it satisfies these conditions; that is, to measure the association of an arbitrary n-gram ($n \geq 2$) and to give higher values to glossary items having high co-occurrence frequencies. Below, $\mid T \mid$ 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.

$$A(T) \;=\; log_{10} f(T) \times \frac{\mid T \mid \cdot f(T)}{\sum_{w_i \in T} f(w_i)} \tag{3}$$

3.5. Discovery and Recovery of Glossary Items

A common distinction made in many of our glossary-based applications is between discovery mode and recovery mode. Discovery mode uses the glossary item discovery operation described in this section. In that mode, the domain-neutral extraction algorithms are applied to the document text to discover lists of glossary items and annotate their occurrences. Recovery mode, on the other hand, refers to an operation in which a relevant set of glossary items (or other items) are already known. These items are supplied, often in the form of an authority list, to an algorithm which can locate (i.e., "recover") and annotate occurrences of those items in documents. In many applications, both discovery and recovery mode can be used together, providing annotations for subsequent processing.

The separation of the operation into two modes is not only architectually expedient, but it also provides a way to enhance the recall of applications that need to identify glossary items. The glossary item discovery subsystem may miss some occurrences of a noun phrase in certain contexts. For instance, an extraction system may fail to recognize "*highly enriched uranium*" as a nominal expression in example (5), because "*has highly enriched*" can be interpreted as a verb group. However, "*highly enriched uranium*" is identified as a nominal expression from other occurrences like (6).

(5) The country has *highly enriched uranium*.
(6) The *highly enriched uranium* recovered from South Africa's nuclear weapons . . .

When an application uses the recovery mode in the pipeline following discovery, the missed occurrences are identified and become available to the application.

4. Experiment and Evaluation

An evaluation for a glossary extraction system can be done in terms of recall and precision by comparing the system's output against a gold standard test set. However, to the best of our knowledge, there is no gold standard test set for the task of glossary extraction or domain-specific terminology identification. Generating a gold standard text for glossary extraction is more challenging than other natural language processing tasks like general NP chunking, which is purely syntactic. Decisions about what constitutes glossary items are, to some extent, subjective and rely on human semantic judgment; the performance of a glossary extraction system is dependent on how a specific application uses glossary items, as described in the next section.

Nevertheless, for evaluation purposes, we treat the glossary extraction task as two separate subtasks—identifying nominal expressions and extracting domain-specific multi-word expressions—and evaluate our method from the two subtask perspectives. First, we examine how well our system recognizes nominal expressions in terms of recall and precision, and then evaluate the quality of final glossary items in terms of domain-specificity of the glossary items.

4.1. Evaluation of Noun Phrase Recognition

The grammars for glossary candidate identification are not designed to perform general noun phrase chunking. However, in this work, we evaluate our nominal expression recognizer as a NP chunking system for the following two reasons. First, since we define glossary items as a subset of base noun phrases (Ramshaw & Marcus, 1995), the performance of the glossary extraction system also depends on its capability for recognizing base noun phrases. Second, there exists a standard test data for NP chunking task.

A standard data set for NP chunking task was built by Ramshaw & Marcus (1995). The data set consists of sections 15–18 of the Wall Street Journal corpus as training material and section 20 of that corpus as test material. We use the test corpus as our gold standard, and measure the performance of our nominal expression recognizer with two scores: precision and recall.

We show the performance of nominal expression recognition using only the syntactic grammars described in section 3.1 and also using those grammars in combination with entity recognizers, as described in section 3.2. In this work, we use two entity recognizers—time expression and company names. The results are shown in Table 2. As we can see in the Table, entity recognizers help to find better glossary items (about 3% increase in both recall and precision).

Table 2
Recall, precision and F-measure of GLOSSEX, compared to Wall Street test data.

	Grammars Only	**Grammars + Entity**
Recall	87.29 %	90.53 %
Precision	85.32 %	88.63 %
F-measure	86.29 %	89.57 %

Major reasons for errors include the following. First, the nominal expression grammars make mistakes in recognizing nominal expressions with conjunctions. For instance, "*his previous real-estate investment and asset-management duties*" was recognized as two nominal expressions–"*his previous real-estate investment*" and "*asset-management duties*". Second, structural information is needed to recognize correct noun phrases. For example, GLOSSEX recognizes "*Iran asylum*" as a nominal expression in "... *had granted the shah of Iran asylum in Panama*". Third, the tagger used in this experiment assigns a wrong part-of-speech to words. For instance, the tagger misinterpreted "*forecast*" as a verb in "*its dismal earnings forecast for 1989*" resulting in "*its dismal earnings*" as a base noun phrase.

Table 3
Sample glossary items extracted from a car users' manual. Abbreviation variants are denoted by all upper case, orthographic variants by italics, misspelling variants by bold; inflectional variants are in plain font.

Canonical Form	**Variants**	**Canonical Form**	**Variants**
anti-lock braking system	ABS, Anti-lock Braking Systems	Supplemental Restraint System	SRS, *Supplemental restraint system*
Overhead Cam	DOHC	transaxle	transaxles
miles per hour	MPH	revolutions per minute	RPM
Single Overhead Cam	SOHC	ignition	
Federal Communications Commission	FCC	child safety seat	*child-safety seat*, *child-safety seats*, child safety seats
corresponding mileage		Sequential multi-port electronic fuel injection	SEFI
Gross Axle Weight Rating	GAWR, Gross Axle Weight Ratings	4-door Windstar	
Overdrive	OD, *over-drive*	seatback	seatbacks
Air Bag	air bags, *airbag*, *airbags*	torque	
lamp	lamps, **lam**	collision	collisions
engine	engines	disc	discs
instrument cluster	*instru-ment cluster*	mode	modes
Gross Combined Weight Rating	GCWR	Gross Combination Weight Rating	GCWR
brake	brakes, braking	powertrain	
Air/Fuel Mixture		frequency band	
Catalytic Converter	Catalytic Converters	windshield	
HAZARD FLASHER	hazard flashers	safety belt	safety belts
rim flange		drivetrain parameter	*drive train parameters*
outer tie-rod	*outer tie rod*	vehicle	vehicles
Ignition Switch		cabin air filter	*cabin air-filter*

4.2. Evaluation of Glossary Items

In this section, we show the performance of our system from the perspective of domain specificity of the extracted glossary items. We conducted the experiment using a document about automobile maintenance. This document file is 1.4 megabytes in length and contains 225,100 words. We extracted terms of up to 6 words in length and their variants from the document and found a total of 9862 glossary items (not including variants). Table 3 above shows the 40 highest confidence glossary items and their variants. For this test run, α was set to 0.1 (see equation 1) for calculating term confidence.

First, we evaluated our system from the perspective of the system's capability for finding domain-specific glossary items. Since there is no standard test set for this task, we did a manual evaluation. It is very time consuming to manually examine a large set of glossary items and to decide if a glossary item is domain-specific or not. Therefore, we reduce the test set to the top 500 glossary items. We presented three human judges with the 500 glossary items and asked them to mark glossary items related to automobiles. Judge 1 marked 303 items; judge 2 marked 299 items; and judge 3 marked 316 items as automobile-related glossary items; that is, in average, 61.2 % of the glossary items are considered domain-specific.

Note that the test document is a car users' manual which explains the functions of, and maintenance information for, the parts an automobile. Some parts of the book describe non-automotive concepts when read separately. For instance, a chapter in the document explains where the radio and the cassette player are located and how to operate them. The judges marked some glossary items found here, such as *"CD mode"*, *"AM/FM Stereo/Cassette"* and *"lamp bulb"* as non-domain-specific.

Second, we compared our metric for computing multi-word term cohesiveness with other well-known metrics for two-word glossary items. Even though our confidence measure provides a unified way to compute confidence values for multi-word terms containing any number of words, we limit the evaluation to two-word terms (or *bigrams*). The reasons are: (1) two-word glossary items are the most common technical terms: in our experiment, 4055 canonical forms among 9862 glossary items are two-word terms; also see (Justeson & Katz, 1995), and (2) there are no generally accepted evaluation mechanisms for terms of more than two words. We compared our term association measure, $A(T)$, with log likelihood ratio (LLR; Dunning 1994) and mutual information (MI; Church & Hanks 1990, which are widely used for extracting word collocations.

For this evaluation, we extracted word pairs of ADJ-NOUN, NOUN-NOUN, and DET-V_PARTICIPLE-NOUN (determiners were removed after recognition) from the test document. Then, we applied three different scoring measures to the extracted bigrams. We counted how many domain-specific terms are included in the top 300 glossary items (T300) and the bottom 200 glossary items (B200) respectively. The evaluation was also done by the three judges. As shown in Table 4, our system generates more good glossary items in the top set and fewer good items in the bottom set than the baseline algorithms.

Table 4
Evaluation for the association measures for two-word glossary items. *GlossEx A(T)* denotes the term cohesion metric in equation 3, LLR denotes log-likelihood ratio, and MI denotes mutual information.

	GlossEx A(T)		LLR		MI	
	T300	B200	T300	B200	T300	B200
Judge1	203	4	162	36	44	39
Judge2	217	7	171	46	56	54
Judge3	228	6	165	48	58	51

5. Applications of Automatic Glossary Extraction

In this section, we examine several applications which use glossary extraction, usingGLOSSEX as a subsystem. We have built these applications using our pipelined text processing infrastructure (Section 2.1) in order to combine glossary extraction functionality with other required text analysis modules and, ultimately, with logic implementing other parts of the applications. This is in addition to the use of the infrastructure for building GLOSSEX itself, as described in Section 3. In fact, some application pipelines explicitly undo filtering that the main glossary extraction algorithm applies—for example by adding proper names and non-nominal expressions to the mix of annotated items expected by downstream application logic. Our descriptions will detail this further use of the infrastructure, where appropriate.

A common theme appears in many of our applications. Although we cannot yet say that systems understand the documents they process, they nevertheless can enhance human understanding by exploiting conceptual characterization of documents' contents. At the core of those characterizations are glossary items, which are the names for domain-specific, and application-specific, concepts and which have the property that they efficiently elicit those underlying concepts when shown to system users. Based on this observation, we employ the general approach of structuring parts of these applications as "semi-automated" processes, in which the system builds the document characterizations automatically but then submits them to human users for final inspection, approval, and/or further use.

5.1. Document Authoring, Indexing, and Search

Customer support and on-line help applications often involve the creation and retrieval of authoritative documents about the products being supported. Among the goals of these systems' administrators are that the documents they contain must be easy to create, easy to find, and easy to understand. In support of these goals, glossary extraction can enable terminological consistency in document authoring, offer comprehensive and meaningful indexing keywords for search, and enhance search operations with query expansion and refinement.

Kozakov *et al.* (2004) describe the use of GLOSSEX in a major corporate technical support application whose knowledge base contains over a million documents. This system exploits semi-automatically-generated glossaries in a number of ways. To begin, the documents were processed with the GLOSSEX, resulting in a candidate glossary consisting entirely of discovered items. The candidate glossary was presented to domain experts by means of a *glossary administration tool* which allowed them to revise the suggested glossary and store the resulting approved glossary. This glossary was used, in recovery mode, to process all the documents in the knowledge base. The salient glossary items found in each document were added to that document's metadata in the knowledge base. The salient items and their variants were also used to index the document in the system's search engine. At the same time, a sentential summary was added to the metadata for each document, with salience values from the glossary being used as part of the summarizer's sentence selection criteria (Boguraev & Neff, 2000). Finally, when the keyword-enhanced search produces a query result, the documents in the hit list are decorated with both their keyword lists and their sentential summaries. In future work, the developers intend to also use the approved glossary as an aid to authors, by showing them preferred glossary items and their variants.

Taxonomies and Categorization. In their evaluation of the technical support application, Kozakov *et al.* (2004) report a potentially useful result involving the interplay between documents' glossary items and the taxonomic categories to which the documents belong. In their system, most of the documents are categorized with respect to a corporate technical taxonomy containing 31 top level nodes. After glossary items for the documents had been assigned as metadata keywords, the authors reported observing that over 90% of all the keywords belonged to documents in a single category. In other words, if a given document is marked with a particular keyword, then it is very likely that another document marked with the same keyword belongs to the same category as the original document. Furthermore, this tendency is even stronger for multi-word glossary items than for single words.

Of course this observation must be studied and the effects of glossary item salience, taxonomy breadth and depth, and other variables must be better understood. Nevertheless, the observation suggests further applications for glossary items. For example, effective classifiers might be built using the glossary contents of documents in a training corpus of already-classified documents. Such classifiers could be used, in turn, in an authoring environment to recommend taxonomic categories to which a newly-written document should be assigned, based on the glossary items the document contains.

5.2. Contact Record Analysis

The number of applications that exploit unstructured text in the operation of contact centers is increasing. The stereotypical contact center is a customer service telephone support operation. Here, the customer service representative (CSR) typically writes a brief summary of the customer call and these collected summaries form the database of contact records. Other types of contact center include: on-line help centers, where customers enter typed questions and CSRs respond with typed answers; field representative call-in centers, where service technicians for, e.g., utility companies phone in to dictate an account of a service incident they've just handled; and dealer service centers, which collect technician-written accounts of service incidents for, e.g., automotive products.

In these applications, the collected contact records constitute a database or data warehouse in which the records contain structured fields, such as date of service, customer ID, or dealer name, along with unstructured text. Among the goals of the applications is to use all the data for managing the contact centers' processes. This can include generating management reports as well as supporting various kinds of and data mining and ad hoc analysis for detecting trends and for problem determination. In order to exploit the contents of the unstructured text in these records, text analysis must be done to impose some structure on it. As with other glossary applications, this one can exploit automatically-generated glossaries to characterize (in discovery mode) the meaningful lexical items in the collected texts and to locate their occurrences (in recovery mode) when individual contact records are being processed.

The unstructured text in contact records has a number of properties affecting its suitability for text analysis. The text is often ungrammatical, is written in nonstandard case, and contains spelling errors. In order to deal with such pathologies, we include special annotators in the text analysis pipeline. For example, we have built specially-trained part-of-speech taggers to deal with monocase text. Of more relevance to our concern with glossary extraction, however, is the fact that they make heavy use of domain-specific jargon, including many abbreviations and acronyms, unknown words, and non-standard spellings of known words. As we have seen, our glossary extraction technology provides mechanisms for dealing with these phenomena.

In one contact center application, devoted to personal computer help centers and discussed in (Nasukawa & Nagano, 2001), administrators create a semantic dictionary, in which glossary items are endowed with category markers such as HARDWARE, SOFTWARE, MACHINE TYPE, PROBLEM, etc. This is done in an off-line application of GLOSSEX and the glossary administration tool. In the subsequent on-line processing pipeline, after the semantic dictionary is used (in recovery mode) to annotate incoming contact records, a part-of-speech tagger and shallow parser (specifically configured to handle GLOSSEX's annotations) are used to assign grammatical relations such as subject-verb (e.g., "*the paint peeled*") and verb-object (e.g., "*install Windows 98*") to the records' text. Next, the information in the resulting annotations of semantic categories and grammatical relations are assigned to the record as metadata in the contact record database. The application uses this metadata in subsequent data mining, trend analysis, and visualization operations.

5.3. Back-of-the-Book Indexing

In the process of publishing technical manuals and other books, a near-final step is devoted to generating an index. Called back-of-the-book indexes, these are built manually today by the book's author, editor, or some other representative of the publisher. The task involves reading the book, identifying the important ideas, choosing concise names to use as index terms for those ideas, and organizing those names into an index, with appropriate page references to the text. Our indexing system automatically generates candidate book indexes. When used in the technical manual generation process, these candidates are presented as a draft index to an editor for final adjustment and approval before publication.

Back-of-the-book index entries consist of index terms, which are most often nominal expressions. Subentries, in multi-level indexes, usually express conceptual relationships involving the main index term and may contain terms which are verbal or adjectival expressions.

Our general approach to automatic indexing begins by broadening the class of nominal expressions extracted, through grammar modification, in order to include items (such as URLs or error message numbers) that are typical of the domain. The resulting glossary items become candidate index entries. We also use recovery mode to apply various authority lists (containing, for example, trademarks or recommended index terms from related books), in order to identify candidate index terms with might not otherwise be properly discovered by GLOSSEX. We further include a shallow parser to recognize relations involving the nominal expressions, for use in generating candidate subentries.

The approach continues by assigning features to the candidate entries and subentries. Although some features have already been assigned by GLOSSEX (e.g., case pattern, frequency, or domain specificity), other features require further text analysis procedures to be included in the pipeline. These new features mark, among others, structural (e.g., "*is it in the table-of-contents or a figure caption?*") and syntactic (e.g., "*is it a subject?*") properties of the candidate index terms' annotations. Finally, scores are computed for the candidate entries, based on their feature values, and the most highly ranked candidates are organized into a draft index for presentation to the human editor.

Initial analysis of our approach indicates that the quality of the generated indexes will rival that of manually-generated ones and that human indexing productivity will be significantly improved. Even without making the adjustments for adapting the base glossary extraction algorithms to the task of candidate index entry identification, we obtained recall rates of 55-65% when comparing automatically-extracted glossaries from books with human-generated index terms for the same books. Furthermore, in related work, Boguraev & Kennedy (1999) report recall and precision in the range of 90% (compared to human performance) when a suitably constructed text analysis pipeline was used to extract indexing terms and relations from an on-line computer help manual. That work differs from ours in that it was based on simple multi-word term identification, supplemented with ad hoc mechanisms for finding single-word candidate index items. Furthermore, it did not benefit from rich notions of domain-specificity assignment to items, such as GLOSSEX provides to our indexing system. Hence, we expect that our eventual evaluation results will be at least as good as those cited by Boguraev & Kennedy (1999).

5.4. Entity Extraction for Ontologies

Recently, applications requiring accurate characterization of the meaning of text in documents have caused a demand for the creation, population, and use of ontologies for expressing those meanings. These needs can range from supporting disambiguation of ambiguous words and phrases—by, e.g., annotating the text with representations of underlying entities—to supporting reasoning over the entities, relationships, and events mentioned in a document.

As we have seen, the nominal expressions discovered by GLOSSEX are an important part of the conceptual characterization of text, since they serve as document mentions of the domain entities. Using this observation, Park (2004) uses GLOSSEX to determine initial entity names, which it subsequently organizes into relationships, as part of an ontology population tool.

In related work, we have enhanced ontology population from text by the addition of more text analysis functions to the GLOSSEX pipeline. For entities, we add recognizers for the proper names, time expressions, and special tokens (e.g., URLs) which the glossary algorithm excludes, if the ontology and/or application require them. Further, to the canonical form and variant mentions that our glossary algorithm identifies, we add other coreferential mentions by including anaphora resolution modules that annotate pronouns and definite noun phrases with correct entity references. Once entities are identified and their mentions annotated, other modules find and annotate mentions of relations and events, aggregate the results, and insert those results into the ontology.

6. Related Work

The idea of using the syntactic shape of nominal expressions as a basis for discovering meaningful referring expressions in unstructured text has its origin in the work reported in (Justeson & Katz, 1995) on technical terminology identification; such ideas have been successfully deployed in applications, for instance in translating techical terminology (Dagan & Church, 1994). A particularly interesting result of Justeson & Katz's research is the observation that syntactic regularities in lexicalized multi-token expressions make terminology identification—with high degree of accuracy—possible on the basis of lexical lookup alone. The downside in their approach is that the nominal expression pattern, while relatively unconstrained by language specificity (Dagan & Church, 1994), is only effective when frequency constraints are applied over multi-token sequences; in other words, the algorithm identifies reliably only multi-token technical terms.

Our work differs from the terminology work by applying more powerful and flexible syntactic analysis tools (e.g., part-of-speech disambiguation and shallow parsing): extensions made necessary by the broader definition of glossary items (as opposed to that of technical terms), and by the need to respond to application- and genre-specifics by swapping grammars. We also develop an effective measure of domain specificity (Section 4.2), which allows us to identify single token glossary items commensurate in quality to longer, multi-token phrases. Additionally, GLOSSEX's more aggressive aggregation produces much richer lists of variants for the extracted items: a natural benefit from a pipelined architecture, where autonomous processes of e.g. abbreviation finding and acronym expansion can be integrated in the algorithm (Section 3.3.2).

Lin (1999) presents a method for automatic identification of non-compositional expressions using their mutual information in a text corpus. The method, which clearly attempts to minimise dependence on linguistic patterns, is based on the hypothesis that when a phrase is non-compositional, its mutual information differs significantly from the mutual information of phrases obtained by substituting one of the word in the phrase with a similar word. The experiments were done on word bigrams which have a VERB-OBJECT, NOUN-NOUN or ADJECTIVE-NOUN relationship. Comparison between mutual information and the term cohesion metric underlying the operation of GLOSSEX has already been discussed in Section 4.2.

Frantzi & Ananiadou (1999) first extract terms from a corpus using the C-Value method which is based on frequency of occurrence, term length and information about whether the term is a substring of other terms. In a following step, they incorporate contextual information in the form of a context factor for each candidate term. A context word is a noun, adjective or verb appearing within a fixed-size window of the candidate term. The context factor of a context word is calculated based on how often the context word appears with the candidate term. The NC-Value of a term is a linear combination of the C-Value and the Context Factor.

Maynard & Ananiadou (2000) extend the NC-Value method of Frantzi & Ananiadou with contextual information of multi-word expressions. Different elements of the context are combined to form the Information Weight (IW), a measure of how strongly related the context is to a candidate term. They decide good terms on the basis of a combination of the NC-Value and IW of terms.

Dias *et al.* (2000) present a n-gram based multi-word extraction technique from a part-of-speech tagged corpus. For each pivot word, they look at a window of six words—three words to the left and three words to the right—and generate all possible, contiguous and non-contiguous, word n-grams: from 2-grams to 7-grams. Cohesiveness for the n-grams is calculated by a Mutual Expectation measure. The Mutual Expectation consists of two factors: the probability of the n-gram in the text and the Normalized Expectation. Similarly to our confidence measure, the Mutual Expectation gives higher scores to the terms which have higher cohesiveness and higher occurrences in the text. Unlike other approaches, which use a global cut-off value to select the final set of multiword terms, they decide an n-gram, W, is a multiword if its association value is higher than or equal to the association values of all $(n-1)$-grams and higher than the association values of all $(n+1)$-grams, which contain W.

7. Conclusion

The paper focuses on a particular class of multiword expressions, namely domain-specific glossary items, and describes a method for glossary identification and extraction which combines, within a flexible operational framework, linguistic insights and observations with statistical measures obtained through empirical analysis of large document collections. The method develops a way of quantifying a basic definitional feature of glossary items: their cohesiveness within a domain. It additionally benefits both from strategies for incorporating domain-specific processing (where, and to the extent appropriate) into a flexibly reconfigurable set of filters, and from the application of strong organizational principles for aggregation of glossary item variants.

A number of experiments with a tool, GLOSSEX, which embodies the algorithm as a scalable system component, demonstrate its robustness across multiple domains, and the high quality of its output, especially where the measure of domain specificity is concerned. This qualifies GLOSSEX as a reliable component to a broad range of sophisticated text processing applications, such as document indexing and search, back-of-the-book indexing, and ontology construction for novel domains.

In order to make a glossary extraction technology like ours available to such a variety of applications, we argue that in addition to making a methodological distinction between discovery and recovery mode of operation of the technology itself, certain architectural features of the text processing infrastructure are required. Such features include the ability to configure (and reconfigure) pipelines of independent system components, whose operation is mediated by a rich and flexible representational model; this makes it possible not only to focus on processes with different granularity (such as, for instance, those of extracting a list of glossary items vs. those of identifying contexts of glossary item use), but also to make dynamic glossary item identification available to an application on demand. To that end, we describe a system architecture which demonstrably supports the integration of GLOSSEX within different applications in various engineering and technical domains. As work continues, we expect that broadening our focus to other domains as well as other types of multiword expressions will exploit our linguistic description capabilities but will not require major modifications to our approach to system integration.

References

Bird, S., and Liberman, M. 2001. A formal framework for linguistic annotation. *Speech Communication* 33(1,2):23–60.

Boguraev, B. K., and Kennedy, C. 1999. Applications of term identification technology: Domain description and content characterisation. *Natural Language Engineering* 5(1):17–44.

Boguraev, B. K., and Neff, M. 2000. Lexical cohesion, discourse segmentation and document summarization. In *Proceedings of RIAO-2000, Content-Based Multimedia Information Access*.

Boguraev, B. 2004. Annotations-based finite state processing in a large scale NLP architecture. In Nicolov, N.; Boncheva, K.; Angelova, G.; and Mitkov, R., eds., *Recent Advances in Natural Language Processing III*, Current Issues in Linguistic Theory . Amsterdam/Philadelphia: John Benjamins.

Church, K., and Hanks, P. 1990. Word association norms, mutual information and lexicography. *Computational Linguistics* 6(1):22–29.

Dagan, I., and Church, K. 1994. *Termight*: Identifying and translating technical terminology. In *Proceedings of 4th Conference on Applied Natural Language Processing*, 34–40.

Damerau, F. 1993. Generating and evaluating domain-oriented multi-word terms from texts. *Information Processing & Management* 29:433–447.

Dias, G.; Guillore, S.; Bassano, J.; and Lopes, J. 2000. Combining linguistics with statistics for multiword term extraction: A fruitful association? In *Proceedings of RIAO2000*.

Dice, L. 1945. Measures of the amount of ecologic associations between species. *Journal of Ecology* 26.

Dunning, T. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics* 19:61–74.

Dunning, T. 1994. Statistical identification of language.

Frantzi, K. T., and Ananiadou, S. 1999. The c-value/nc-value domain independent method for multi-word term extraction. *Journal of Natural Language Processing* 6(3):145–179.

IBM. 2001. IBM dictionary and linguistic tools.

Justeson, J., and Katz, S. 1995. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural Language Engineering* 1:9–27.

Karttunen, L.; Chanod, J.-P.; Grenfenstette, G.; and Schiller, A. 1996. Regular expressions for language engineering. *Natural Language Engineering* 4(1):305–328.

Kinyon, A. 2001. A language-independent shallow-parser compiler. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*.

Kozakov, L.; Park, Y.; Fin, T.; Drissi, Y.; Doganata, Y.; and Cofino, T. 2004. Glossary extraction and utilization for ibm technical support information search and delivery system. *IBM Systems Journal* 43(3).

Levenshtein, V. 1962. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* 10(8):707–710.

Lin, D. 1999. Automatic identification of non-compositional phrases. In *Proceedings of ACL-99*, 317–324.

Maynard, D., and Ananiadou, S. 2000. Identifying terms by their family and friends. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Nasukawa, T., and Nagano, T. 2001. Text analysis and knowledge mining system. *IBM Systems Journal* 40(4):967–984.

Neff, M.; Byrd, R.; and Boguraev, B. K. 2004. The Talent system: TEXTRACT architecture and data model. *Natural Language Engineering* 10(4). Special Issue on *Software Architectures for Language Engineering*.

Park, Y., and Byrd, R. J. 2001. Hybrid text mining for matching abbreviations and their definitions. In *Proceedings of Empirical Methods in Natural Language Processing*, 126–133.

Park, Y.; Byrd, R. J.; and Boguraev, B. K. 2002. Automatic glossary extraction: Beyond terminology identification. In *Proceedings of the Nineteenth International Conference on Computational Linguistics (COLING02)*.

Park, Y. 2002. Identification of probable real words: An entropy-based approach. In *Proceedings of ACL Workshop on Unsupervised Lexical Acquisition*, 1–8.

Park, Y. 2004. Glossont: A concept-focused ontology building tool. In *Proceedings of Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR2004)*.

Ramshaw, L. A., and Marcus, M. P. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*.

Smadja, F. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics* 19(1):143–177.