Automatic Glossary Extraction: Beyond Terminology Identification

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

Abstract

This paper describes a method for automatically extracting domain-specific glossaries from large document collections. We show that, compared with current text analysis methods for extracting technical terminology from text, our extracted glossaries more successfully support applications requiring knowledge of domain concepts. After presenting our methods, we illustrate the output of GlossEx, our glossary extraction tool, and present an informal evaluation of its performance.

1 Introduction

In recent years, significant progress has been made in using text analysis methods to extract information from document text, with the ultimate goal of supporting a variety of applications that rely on document meaning. Starting with the simplest word-based feature extractors (see, for example, Baeza-Yates and Ribeiro-Neto, 1999), these methods are maturing into a rich set of symbolic and numerical methods (as, for example, in Rindflesch2000) for identifying and organizing text segments that correspond to human conceptual understanding of the texts' meaning.

This paper describes glossary extraction, an important component of text understanding systems. The objective of glossary extraction is to identify and organize words and phrases from documents into sets of "glossary items". Domain-specific glossaries, consisting of items drawn from large collections of documents for particular domains, contain an account of the important conceptual material in the domains. Specifically, glossary items name and describe the domain concepts in a way that can be exploited by applications. The items can consist of canonical and variant forms of the concepts' names, syntactic information about the forms, definitions of the underlying concepts, and relationships that link concepts.

Glossary items are different from technical terms as described, for example, by Justeson and Katz (1995). In the present paper, we will show that glossary items result from the identification of single-word forms, abbreviations, verbs, and salience. The techniques used to make these identifications include part-of-speech tagging and parsing, induction and application of abbreviation rules, aggregation of multiple forms, and statistical computations of item distribution. These all go significantly beyond the manipulation of noun phrases typically used for term identification. Nevertheless, the principles of term identification play an important role in the recognition of forms of which glossary items may consist.

In glossary-based applications, we use the following three-part scenario. We begin with glossary extraction to automatically extract candidate items from a large collection of documents from the application domain. Next, using a glossary administration system, we present the candidate glossary to a domain expert or librarian for review, modification, and approval. The final approved glossary is made available, through suitable APIs, to the application system.

Applications for domain-specific glossaries range from those that support direct human use to those that address the needs of computers. Human use is supported by published glossaries, on-line glossary reference tools, and authoring environments that use glossaries to enable or enforce terminological consistency. An example we've seen is in an engineering environment where glossary items are the only approved names of features in CAD drawings. Computers can use glossaries for document indexing and search, federation of heterogeneous document collections, document summarization and keyword extraction, and automated construction of domain taxonomies and ontologies. Many of these applications appear in knowledge portals (Mack et al. , 2001).

Glossary extraction is one application of Textract, the text analysis system being developed by the Text Analysis and Language Engineering ("Talent") project at IBM Research. In Textract, text analysis is performed by flexible configurations of analyzers (also called "annotators" or "plugins") which interact by communicating their various analyses of a document's text through annotations. Textract currently contains some tens of analyzers for functions such as tokenization, lexical lookup and morphological analysis (IBM Research, 2001), named-entity identification, part-of-speech tagging, shallow parsing, anaphora resolution, topic segmentation, and relation extraction. Talent applications, also built from configurable Textract plugins, use the resulting annotations for various purposes such as summarization (Boguraev and Neff, 2000), lexical network extraction (Cooper and Byrd, 1997), document collection indexing, as well as glossary extraction. Our approach to building text analyzers allows us to achieve several crucial goals: through reconfiguration, we can address diverse applications; by using shallow and finite-state methods, 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.

The remainder of the paper details the glossary extraction process. We first identify candidate glossary item forms, by recognizing specific syntactic structures in POS-tagged text (section 2.2). Second, we filter generic (i.e, non-domain-specific) pre-modifiers from the candidate forms (section 2.3). Third, we aggregate forms that are variant names for the same concept into candidate glossary items (section 3). Finally, we use statistical information to rank candidate items and to compute confidence values which can be used for subsequent filtering of the candidate glossary (section 4). Section 5 presents sample output from glossary extraction, along with an informal evaluation of its accuracy. Section 6 concludes the paper and describes ongoing work based on glossary extraction.

2 A Glossary Extraction Algorithm

We identify single-word and multi-word glossary items in text. Most terminology extraction systems (Dias et al. , 2000; Damerau, 1993; Jacquemin, 1995) have been focused only on extracting multi-word noun phrases. However, verbs and single-word nouns in domain-specific documents also contain domain information. We also pay attention to out-of-vocabulary words [1] because most technical jargon is not likely to be included in a general-purpose dictionary.

2.1 Structure of Glossary Items

A glossary item is either a noun phrase or a verb in this work (see, Justeson and Katz, 1995, for more detail description on the structure of technical terminology). For verbs, we consider only non-auxiliary verbs and take their base forms as candidate glossary items. For noun phrases, we derived the structure of noun phrases based on the study by Justeson and Katz (1995), and domain experts' analysis on the documents in one of our experiment domains. Figure 1 shows how noun phrases are recognized.

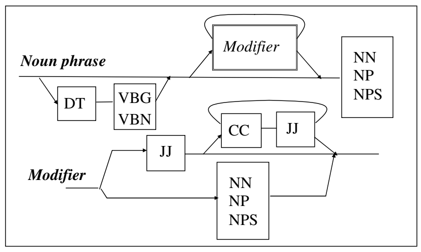

Figure 1: Structure of Noun Phrases

In this figure, lexical units are represented by Penn Treebank Tag codes (Marcus and Santorini, 1993), where DT denotes a determiner, VBG and VBN denote participle forms of verbs, JJ denotes an adjective, CC denotes a conjunction, and NN, NP, NPS denote nouns.

2.2 Identification of Candidate Glossary Items

A common strategy for technical terminology identification, given the relatively simple "contour" of a term noun phrase, is to detect such phrases by means of lexical, part-of-speech driven, filtering (see Justeson and Katz, 1995, for an argument in favour of such an approach). The identification of candidate glossary items, however, poses challenges which cannot be met by a simple lexical filter. For one thing, the noun phrase structure defined in Figure 1 is more complex than that of a term phrase, and the length of a glossary item does regularly exceed the average length of technical terms (even without accepting trailing prepositional phrases. Greater length makes a recognizer more susceptible to part-of-speech ambiguities. In addition, identification of verbal glossary items is equally open to problems of ambiguity, which cannot be resolved by lexical look-up alone.

This leads us to adopt a "pipeline" architecture, where look-up, morphological analysis, and part-of-speech tagging run as pre-processors to the glossary extraction process. Note that such an architecture allows for natural "insertion" into the pipeline of additional filters (such as named entity or URL recognition; see below) prohibiting certain phrases which fit the contour of Figure 1 from being considered as good glossary item candidates. The grammar sketched in Figure 1 is derivative of a larger grammar for noun phrases, itself a part of a shallow parser (for English) realized as a cascade of finite-state (FS) transducers (Boguraev, 2000).

There are a number of reasons for using FS technology for a task like the one here (see, for in-

[1] words which are not found in a dictionary

stance, Karttunen et al., 1996; Kornai, 1999). We are particularly concerned with issues of re-usability of grammars (in the sense just described, where a general-purpose NP grammar with known properties and coverage is trimmed/modified in an adaptation for a narrower range of patterns), adaptability 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 changes), 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, 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 (see Kinyon, 2001, for compiling language-independent "chunkers").

Our pipeline architecture requires a special-purpose FS parsing regime. In order to be able to integrate, seamlessly, a number of pre-processors and filters in a re-configurable environment for document collection processing, the results of linguistic analysis of any type are couched in terms of posting annotations over text spans. Noun phrases are just one example of such annotations; in principle, there is no a priori limit on the annotations' number and types: words, named entities, special tokens, phrases, chunks, sentences, and so forth are all annotations. While 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 characters. We have developed a generalization of the notion of character-based finite state transduction, in order to be able to define patterns in terms of annotations, and have a finite-state executor read from, and write to, an annotation repository. The details of this FS model are outside the scope of this paper, but we will note here that in general, this approach not only improves, substantially, the efficiency of an FS-based recognizer, but also makes it very natural to implement finite-state cascades, 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.

We discard some kinds of the recognized forms from the candidate set. These forms are as follows:

- Forms having more than 6 words
- Person names and place names (Ravin et al. , 1997)
- Special tokens such as URLs and words containing special symbols except hyphens and dashes

Some examples of candidate glossary item forms are shown in Table 1. In this example, A denotes adjectives, N denotes nouns, and C denotes conjunctions.

Structure	Terms
AN	genuine part
NN	sport utilities
AAN	heavy commercial use
ANN	rear wiper blade
NNN	emission control system
AANN	other qualified service technician
ACAN	unpaved or dusty roads
ANNN	automatic transmission fluid level
NNNN	engine oil fluid level
AANNN	new personalized oil reset percentage
AACAN	certain frontal or near-frontal collision
ACAAN	ambient and wide open throttle
NNNNN	steering wheel fan speed control

Table 1: Examples of candidate forms

2.3 Pre-modifier Filtering

Many pre-modifiers in noun phrases, even in domain-specific noun phrases, act as general-purpose modifiers rather than representing domain-specific information. There are two problems in including all pre-modifiers 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 pre-modifier, we remove it from the noun phrase and take the remaining noun phrase as a candidate form.

How can we decide if a pre-modifier is domain-specific or generic? The easiest way might be to keep a list of generic pre-modifiers and to remove them from candidate forms. However, some pre-modifiers may be domain-specific in one domain but not in others. So, when the domain changes, the pre-modifier list would need to be changed as well. Collecting a list of generic pre-modifiers for each domain is labor-intensive and error-prone. In this work, we automatically decide whether a pre-modifier should be filtered based on the domain-specificity of the pre-modifier and the association of the pre-modifier with the noun it modifies. We regard the first noun after a pre-modifier as the modified noun. Domain-specificity means how much a word represents information about the domain and is determined by the relative probability of the occurrence of the word in a domain-specific document and in a general corpus. Domain-specificity is described in more detail in section 4. The association of a pre-modifier (a) and a noun (n) is calculated by the conditional probability, $p(n|a)$, of the occurrence of the head noun given the pre-modifier (see, Lapata et al., 1999 for further discussion on adjective-noun plausibility) .

The decision process is shown in Figure 2. If a pre-modifier's domain-specificity (D) is very low (less than a low threshold ε), then we remove the pre-modifier unless the association (A) is very strong (greater than an upper threshold α). On the other hand, if a pre-modifier's domain-specificity is very strong (greater than an upper threshold ζ), then we take the pre-modifier unless the association is very low (less than a threshold β). If the domain-specificity is in-between ε and ζ, then we base the decision on the association of a pre-modifier and a noun. If the association is greater than a threshold, γ, then we keep the pre-modifier, otherwise we discard it. For the experiment, we set α to 0.3, β to 0.005, γ to 0.02, ε to 1.5, and ζ to 3.0.

```
If (D < $\varepsilon$ and A < $\alpha$)
    remove pre-modifier
else if (D ≥ $\zeta$ and A ≥ $\beta$)
    take pre-modifier
else if (A ≥ $\gamma$)
    take pre-modifier
else remove pre-modifier
```

Figure 2: Pre-modifier Filter

Upon application of this decision process, the pre-modifiers *other*, *new* and *certain* are removed from noun phrases *other qualified service technician*, *new personalized oil reset percentage* and *certain frontal or near-frontal collision* respectively. However, *unpaved or dusty roads* and *ambient and wide open throttle* are not affected by this process.

3 Variant Aggregation

A single concept may be represented in different ways in text – for example, as misspellings or abbreviations. We identify conceptually identical expressions among candidate glossary item forms and aggregate them into one glossary item. We select one of the forms as the canonical form and make the other forms its variants. We also combine the frequencies of all different forms so that terms with many variant occurrences may have higher confidence values. We describe how to calculate the confidence value of terms in section 4. The different forms we take into account are:

- Symbolic Variants
- Compounding Variants
- Inflectional Variants
- Misspelling Variants
- Abbreviations

3.1 Symbolic Variants

If two forms are composed of exactly the same words but have different separators, such as hyphens or dashes, we consider them to be variants of one another. The form with higher frequency is selected as the canonical form. Examples of this situation are:

- *audio/visual input - audio-visual input*
- *electro-magnetic clutch - electromagnetic clutch*

3.2 Compounding Variants

People tend to use both compound form and separated form freely and both of the two forms are often used in a document. The term with higher frequency is selected as the canonical form. If the two forms have a same frequency, the separated form is chosen as the canonical form. *Passenger Airbag* and *passenger air bag*, *Mass Airflow Sensor* and *Mass Air Flow Sensor* are recognized as compounding variants.

3.3 Inflectional Variants

This is the most common variant type. For single-word terms, we recognize all the inflectional forms found in the given text. For example, *rewinds*, *rewinding*, and *rewound* are recognized as variants for *rewind*. For multi-word terms, we find same terms but only the last words are inflected. We use a lexical analysis system in Talent (IBM Research, 2001) to obtain lexical information. An example of this case is *parking brake pedal pad* and *parking brake pedal pads*. *Fog lamps* and *foglamps* are identified as variant forms of *Fog lamp* by the compounding variant finding and the inflectional variant finding.

3.4 Misspelling Variants

We recognize misspelled words and their correct forms in text, and link them together as a same glossary item. In order to identify misspelled words and their correct forms, we use a spell-aid function in IBM Dictionary and Linguistic Tools (IBM, 2001) and a string edit distance mechanism. For each out-of-vocabulary word, we run the spell-aid function to obtain possible correct forms for the misspelled word. It returns up to 6 possible correct words from its dictionary. We compute the string edit distance for the misspelled word and each of the candidate correct forms. If the edit distance is small (less than 2) and a candidate correct form appears in the document, we take the correct form as the canonical form with the misspelling as its variant. For example, *accelarator* is found as a misspelling variant of *accelerator*.

3.5 Abbreviations

Documents, especially technical documents, contain many abbreviated forms and the proper recognition of abbreviations and their definitions is important for understanding the documents and for extracting information from them. We use the method described in (Park and Byrd, 2001), for recognizing abbreviations and their definitions in text. Examples of the method's output are:

• 4H	:	four-wheel drive high
• ATF	:	automatic transmission fluid
• DOHC	:	Double Overhead Cam
• NVH	:	noise, vibration and harshness

4 Glossary Item Ranking and Selection

Having obtained possible candidate items, we rank them in order to select the final set of glossary items. In order to rank all glossary items together regardless of their lengths, we need one measure to judge their goodness. We decide the goodness of each term based on how much an item is related to the given domain, the item's domain-specificity, and the degree of association of all words in the item's canonical form (hereinafter *term*). The latter quantity we call term cohesion. That is, the term confidence of a term $T, C(T)$, is defined as in equation 1.

$$C(T) = \alpha * TD(T) + \beta * TC(T) \qquad (1)$$

where TD is term domain-specificity, TC is term cohesion, and α and β (α+ β=1) are constant values which decide the relative contributions of TD and TC respectively.

4.1 The degree of domain-specificity

If an item is used more often in a domain-specific document than in other document collections, it is likely a domain-specific term. We evaluate the domain-specificity of a word based on the relative probability of the occurrence of the word in the given domain-specific text and in a general corpus. Relative frequency ratios are effective to find subject-specific collocations (Damerau, 1993). We define the domain-specificity of a multi-word term as the average of the domain-specificity of all the words in the term as in equation 2.

$$\frac{\sum_{w_i \in T} \frac{P_d(w_i)}{P_c(w_i)}}{\mid T \mid} \qquad (2)$$

where, $\mid T \mid$ is the number of words in term 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. The probabilities are estimated by the frequencies normalized by the size of the corpus.

4.2 The degree of term cohesion

Many methods for evaluating term association have been proposed in other work (Church and Hanks, 1990; Damerau, 1993; Dunning, 1993; Schone and Jurafsky, 2001). However, these measures have two major drawbacks. First, they evaluate the degree of association between two units and need to apply special techniques to calculate the association of terms with more than two words (Dias et al. , 2000). Second, these measures tend to give higher values for low frequency terms, especially mutual information.

In this work, we propose a new measure to compute the cohesion of multi-word terms. Our goal is to measure the association of an arbitrary n-gram ($n \geq 2$), and to give higher values to terms having high co-occurrence frequencies. We generalized the Dice coefficient (Dice, 1945; Schone and Jurafsky, 2001) so that it satisfies our two goals as in equation 3. The measure is proportional to the co-occurrence frequency and the length of the term.

$$\frac{\mid T \mid \times log_{10} f(T) \times f(T)}{\sum_{w_i \in T} f(w_i)} \qquad (3)$$

where, $\mid T \mid$ 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. The equation 3 produces much higher values for single-word terms than multi-word 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%).

Table 2 illustrates the difference of our measure and Dice coefficient for two-unit terms. In this table, *f1* and *f2* denote the frequencies of the first and second units respectively, and *f12* denotes the frequency of their co-occurrences. Dice coefficient produces 1 (perfect association) regardless of their co-occurrence frequencies if the two units always appear together. However, our measure generates association values in proportional to their co-occurrence frequencies.

f1	*f2*	*f12*	*Dice*	*Proposed*
3	2	2	0.8	0.24
10	10	10	1	1
30	20	20	0.625	1.04
100	100	100	1	2

Table 2: Comparison of two association measure

5 Experiments and Results

5.1 Sample Glossary Items

We conducted experiments on the proposed method using a large document about automobile maintenance. This document file is 1.4 megabytes and contains 225,100 words. We extracted up to 6 word terms and their variants from the document. We found a total of 9862 glossary items (not including variants). Table 3 shows the highest confidence 45 glossary items and their variants in parentheses. We set α to 0.1 and β to 0.9 (equation 1) for calculating term confidence.

anti-lock braking system (ABS, Anti-lock Braking Systems)
revolutions per minute (RPM)
Overhead Cam (DOHC)
Sequential multi-port electronic fuel injection (SEFI)
miles per hour (mph)
Supple-mental Restraint System (SRS, Supplemental restraint system)
Single Overhead Cam (SOHC)
ignition ()
Federal Communications Commission (FCC)
vehicle (vehicles)
corresponding mileage ()
transaxle (transaxles)
Gross Axle Weight Rating (GAWR, Gross Axle Weight Ratings)
Windstar ()
LIFTGATE ()
Overdrive (OD, over-drive)
seatback (seatbacks)
Air Bag (Air bags, airbag, airbags)
torque ()
lamp (lamps, lam)
collision (collisions)
equip (equipped)
tow (Towing, towed, tows)
engine (engines)
activate (activated, activating, activates)
disc (discs)
Motorcraft ()
mode (modes)
powertrain ()
Gross Combination Weight Rating (GCWR)
brake (brakes, braking)
illuminate (illuminates, illuminated)
Gross Combined Weight Rating (GCWR)
BELT (belts, belted)
frequency band ()
Catalytic Converter (Catalytic Converters)
windshield ()
feature (features, feauture)
deactivate (deactivated, deactivating,deactivates)
driver (drivers)
roadside emergency (Roadside emergencies)
POWERTRAINS ()
fuse (fuses)
rotate (rotating, rotates, rotated)
axle (axles)

Table 3: Sample Glossary Items

5.2 Performance Evaluation

In order to assess the performance of the proposed method, we have conducted two evaluations. First, we evaluated our system by counting good domain-specific glossary items from the top 500 glossary items. Three judges inspected the 500 glossary items and marked good glossary items. Judge 1 marked 303 items; judge 2 marked 299 items; and judge 3 marked 316 items as automobile-related glossary items

Second, we compared our ranking measure with other well-known metrics for two-word glossary items. The reasons we choose two-word items 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, and also see, Justeson and Katz, 1995), and (2) there are no generally accepted evaluation mechanisms for more than two word terms. We compared our two-word terms with the results from using log likelihood ratio (LLR) (Dunning, 1993) and mutual information (MI) (Church and Hanks, 1990), which are widely used for extracting word collocations.

For this evaluation, we extracted word pairs of adjective-noun, noun-noun and determiner-verb participle-noun (determiners were removed after the 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.

	GlossEx		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

Table 4: Evaluation for Two-word Glossary Items

6 Conclusion and Future Work

We have developed an approach to constructing domain-specific glossaries through text analysis of large document collections. Our techniques go well beyond those used for terminology identification and entity extraction

We have used these methods to build GlossEx, a glossary extraction tool. GlossEx has been used to build glossaries for applications in the automotive engineering and computer help desk domains. Together with a glossary administration tool, not described here, GlossEx has been deployed in applications which process gigabytes of text, yielding tens of thousands of glossary items, and serving large user communities. As Textract's multilingual capabilities develop, we will also build non-English versions of GlossEx. Working with customers and colleagues, we will expand the set of domains addressed, starting with medicine and bioinformatics.

Finally, we plan to use glossary extraction as the basis for a continuing effort in domain ontology construction. New analysis capabilities in GlossEx will include definition extraction, anaphora resolution, and identification of synonyms and other ontological relations. Building on our conviction that domain texts are a rich source of domain knowledge, we also look forward to adapting the ontologies we extract for use with more formal knowledge representation and inference systems.

References

R. Baeza-Yates and B. Ribeiro-Neto. 1999. Modern Information Retrieval. Addison Wesley.

B. Boguraev. 2000. Towards Finite-State Analysis of Lexical Cohesion. In *Proceedings of the 3rd International Conference on Finite-State Methods for NLP, INTEX-3*, Liege, Belgium.

B. Boguraev and M. Neff 2000. Lexical Cohesion, Discourse Segmentation and Document Summarization. In *Proceedings of RIAO-2000*

K. Church, and P. Hanks. 1990. Word Association norms, nutual information and lexicography. *Computational Linguistics* 6(1), pp. 22-29.

J. Cooper and R. Byrd. 1997. Lexical Navigation - Visually Prompted Query Expansion and Refinement. In *Proceedings of DIGLIB'97.*

G. Dias. S. Guillore., J. Bassano, and J. Lopes. 2000. Combining Linguistics with Statistics for Multiword Term Extraction: A Fruitful Association?. In *Proceedings of RIAO2000.*

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

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

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

The Eagles Lexicon Interest Gruop. 1996. Preliminary Recommendations on Semantic Encoding Interim Report. http://www.ilc.pi.cnr.it/EAGLES96/rep2/.

J. Hobbs, D. Appelt, J. Bear, D. Israel, M. Kameyama, M. Stickel and M. Tyson. 1997. FASTUS:A Cascaded Finite-State Transducer for Extracting Information from Natural-Language Text. In *Finite-State Language Processing* pp 383-406, Roche E. and Y. Schabes, eds., The MIT Press, Cambridge, MA.

IBM. 2001. IBM Dictionary and Linguistic Tools. *http://booksrv1.raleigh.ibm.com/lingtool.*

IBM T. J. Watson Research. 2001. The Talent (Text Analysis and Language Engineering) project. *http://www.research.ibm.com/talent/.*

C. Jacquemin. 1995. A Symbolic and Surgical Acquisition of terms through Variation. In *Proceedings of Workshop New Approaches to Learning for NLP* at 14th IJCAI'95.

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

L. Karttunen, J. Chanod, G. Grenfenstette, and A. Schiller. 1996. Regular Expressions for Language Engineering. *Natural Language Engineering*, 4(1), pp.305-328.

A. Kinyon. 2001. A Language-Independent Shallow-Parser Compiler. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France.

A. Kornai. 1999. *Extended Finite State Models of Language*, Cambridge University Press, Cambridge, UK

M. Lapata, S. McDonald, and F. Keller. 1999. Determinants of Adjective-Noun Plausibility. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pp 30-36.

R. Mack, Y. Ravin, and R. Byrd. Knowledge portals and the emerging digital knowledge workplace. In *IBM Systems Journal*, vol. 40, no. 4.

D. Maynard and S. Ananiadou. 1999. *Term Extraction using a Similarity-based Approach*, John Benjamins.

M. Marcus and M. Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2), pp313-330.

Y. Park and R. Byrd. 2001. Hybrid Text Mining for Matching Abbreviations and their Definitions. In *Proceedings of Empirical Methods in Natural Language Processing*, pp 126-133 .

Y. Ravin, N. Wacholder, and M. Choi. 1997. Disambiguation of proper names in text. *17th Annual ACM-SIGIR Conference.*

T. Rindflesch, L. Tanabe, J. Weinstein, and L. Hunter. 2000. EDGAR: Extraction of Drugs, Genes and Relations from the Biomedical Literature. In *Proceedings of the Pacific Symposium on Biocomputing.*

P. Schone and D. Jurafsky, D. 2001. Is Knowledge-Free Induction of Multiword Unit Dictionary Headwords a Solved Problem?. In *Proceedings of Empirical Methods in Natural Language Processing*, pp 100-108 .