# Linguistically Motivated Features for Enhanced Back-of-the-Book Indexing

**Andras Csomai** and **Rada Mihalcea**
Department of Computer Science
University of North Texas
csomaia@unt.edu,rada@cs.unt.edu

## Abstract

In this paper we present a supervised method for back-of-the-book index construction. We introduce a novel set of features that goes beyond the typical frequency-based analysis, including features based on discourse comprehension, syntactic patterns, and information drawn from an online encyclopedia. In experiments carried out on a book collection, the method was found to lead to an improvement of roughly 140% as compared to an existing state-of-the-art supervised method.

## 1 Introduction

Books represent one of the oldest forms of written communication and have been used since thousands of years ago as a means to store and transmit information. Despite this fact, given that a large fraction of the electronic documents available online and elsewhere consist of short texts such as Web pages, news articles, scientific reports, and others, the focus of natural language processing techniques to date has been on the automation of methods targeting short documents. We are witnessing however a change: more and more books are becoming available in electronic format, in projects such as the Million Books project (http://www.archive.org/details/millionbooks), the Gutenberg project (http://www.gutenberg.org), or Google Book Search (http://books.google.com). Similarly, a large number of the books published in recent years are often available – for purchase or through libraries – in electronic format. This means that the need for language processing techniques able to handle very large documents such as books is becoming increasingly important.

This paper addresses the problem of automatic back-of-the-book index construction. A back-of-the-book index typically consists of the most important keywords addressed in a book, with pointers to the relevant pages inside the book. The construction of such indexes is one of the few tasks related to publishing that still requires extensive human labor. Although there is a certain degree of computer assistance, consisting of tools that help the professional indexer to organize and edit the index, there are no methods that would allow for a complete or nearly-complete automation.

In addition to helping professional indexers in their task, an automatically generated back-of-the-book index can also be useful for the automatic storage and retrieval of a document; as a quick reference to the content of a book for potential readers, researchers, or students (Schutze, 1998); or as a starting point for generating ontologies tailored to the content of the book (Feng et al., 2006).

In this paper, we introduce a supervised method for back-of-the-book index construction, using a novel set of linguistically motivated features. The algorithm learns to automatically identify important keywords in a book based on an ensemble of syntactic, discourse-based and information-theoretic properties of the candidate concepts. In experiments performed on a collection of books and their indexes, the method was found to exceed by a large margin the performance of a previously proposed state-of-the-art supervised system for keyword extraction.

## 2 Supervised Back-of-the-Book Indexing

We formulate the problem of back-of-the-book indexing as a supervised keyword extraction task, by making a binary yes/no classification decision at the

level of each candidate index entry. Starting with a set of candidate entries, the algorithm automatically decides which entries should be added to the back-of-the-book index, based on a set of linguistic and information theoretic features. We begin by identifying the set of candidate index entries, followed by the construction of a feature vector for each such candidate entry. In the training data set, these feature vectors are also assigned with a correct label, based on the presence/absence of the entry in the gold standard back-of-the-book index provided with the data. Finally, a machine learning algorithm is applied, which automatically classifies the candidate entries in the test data for their likelihood to belong to the back-of-the-book index.

The application of a supervised algorithm requires three components: a data set, which is described next; a set of features, which are described in Section 3; and a machine learning algorithm, which is presented in Section 4.

### 2.1 Data

We use a collection of books and monographs from the eScholarship Editions collection of the University of California Press (*UC Press*),[1] consisting of 289 books, each with a manually constructed back-of-the-book index. The average length of the books in this collection is 86053 words, and the average length of the indexes is 820 entries. A collection of 56 books was previously introduced in (Csomai and Mihalcea, 2006); however, that collection is too small to be split in training and test data to support supervised keyword extraction experiments.

The UC Press collection was provided in a standardized XML format, following the Text Encoding Initiative (TEI) recommendations, and thus it was relatively easy to process the collection and separate the index from the body of the text.

In order to use this corpus as a gold standard collection for automatic index construction, we had to eliminate the *inversions*, which are typical in human-built indexes. *Inversion* is a method used by professional indexers by which they break the ordering of the words in each index entry, and list the head first, thereby making it easier to find entries in an alphabetically ordered index. As an example, consider the entry *indexing of illustrations*, which, following inversion, becomes *illustrations, indexing of*. To eliminate inversion, we use an approach that generates each permutation of the composing words for each index entry, looks up the frequency of that permutation in the book, and then chooses the one with the highest frequency as the correct reconstruction of the entry. In this way, we identify the form of the index entries as appearing in the book, which is the form required for the evaluation of extraction methods. Entries that cannot be found in the book, which were most likely generated by the human indexers, are preserved in their original ordering.

For training and evaluation purposes, we used a random split of the collection into 90% training and 10% test. This yields a training corpus of 259 documents and a test data set of 30 documents.

### 2.2 Candidate Index Entries

Every sequence of words in a book represents a potential candidate for an entry in the back-of-the-book index. Thus, we extract from the training and the test data sets all the n-grams (up to the length of four), not crossing sentence boundaries. These represent the candidate index entries that will be used in the classification algorithm. The training candidate entries are then labeled as positive or negative, depending on whether the given n-gram was found in the back-of-the-book index associated with the book.

Using a n-gram-based method to extract candidate entries has the advantage of providing high coverage, but the unwanted effect of producing an extremely large number of entries. In fact, the resulting set is unmanageably large for any machine learning algorithm. Moreover, the set is extremely unbalanced, with a ratio of positive and negative examples of 1:675, which makes it unsuitable for most machine learning algorithms. In order to address this problem, we had to find ways to reduce the size of the data set, possibly eliminating the training instances that will have the least negative effect on the usability of the data set.

The first step to reduce the size of the data set was to use the candidate filtering techniques for unsupervised back-of-the-book index construction that we proposed in (Csomai and Mihalcea, 2007). Namely, we use the commonword and comma filters, which are applied to both the training and the test collections. These filters work by eliminating all the n-grams that begin or end with a common word (we use a list of 300 most frequent English words), as well as those n-grams that cross a comma. This results in a significant reduction in the number of neg-

[1] http://content.cdlib.org/escholarship/

| | positive | negative | total | positive:negative ratio |
|---|---|---|---|---|
| | | | Training data | |
| All (original) | 71,853 | 48,499,870 | 48,571,723 | 1:674.98 |
| Commonword/comma filters | 66,349 | 11,496,661 | 11,563,010 | 1:173.27 |
| 10% undersampling | 66,349 | 1,148,532 | 1,214,881 | 1:17.31 |
| | | | Test data | |
| All (original) | 7,764 | 6,157,034 | 6,164,798 | 1:793.02 |
| Commonword/comma filters | 7,225 | 1,472,820 | 1,480,045 | 1:203.85 |

Table 1: Number of training and test instances generated from the UC Press data set

ative examples, from 48 to 11 million instances, with a loss in terms of positive examples of only 7.6%.

The second step is to use a technique for balancing the distribution of the positive and the negative examples in the data sets. There are several methods proposed in the existing literature, focusing on two main solutions: undersampling and oversampling (Weiss and Provost, 2001). Undersampling (Kubat and Matwin, 1997) means the elimination of instances from the majority class (in our case negative examples), while oversampling focuses on increasing the number of instances of the minority class. Aside from the fact that oversampling has hard to predict effects on classifier performance, it also has the additional drawback of increasing the size of the data set, which in our case is undesirable. We thus adopted an undersampling solution, where we randomly select 10% of the negative examples. Evidently, the undersampling is applied only to the training set.

Table 1 shows the number of positive and negative entries in the data set, for the different preprocessing and balancing phases.

## 3 Features

An important step in the development of a supervised system is the choice of features used in the learning process. Ideally, any property of a word or a phrase indicating that it could be a good keyword should be represented as a feature and included in the training and test examples. We use a number of features, including information-theoretic features previously used in unsupervised keyword extraction, as well as a novel set of features based on syntactic and discourse properties of the text, or on information extracted from external knowledge repositories.

### 3.1 Phraseness and Informativeness

We use the phraseness and informativeness features that we previously proposed in (Csomai and Mihalcea, 2007). Phraseness refers to the degree to which a sequence of words can be considered a phrase. We use it as a measure of lexical cohesion of the component terms and treat it as a collocation discovery problem. Informativeness represents the degree to which the keyphrase is representative for the document at hand, and it correlates to the amount of information conveyed to the user.

To measure the **informativeness** of a keyphrase, various methods can be used, some of which were previously proposed in the keyword extraction literature:

- *tf.idf*, which is the traditional information retrieval metric (Salton and Buckley, 1997), employed in most existing keyword extraction applications. We measure inverse document frequency using the article collection of the online encyclopedia Wikipedia.

- $\chi^2$ *independence test*, which measures the degree to which two events happen together more often than by chance. In our work, we use the $\chi^2$ in a novel way. We measure the informativeness of a keyphrase by finding if a phrase occurs in the document more frequently than it would by chance. The information required for the $\chi^2$ independence test can be typically summed up in a contingency table (Manning and Schutze, 1999):

| count(phrase in document) | count(all other phrases in document) |
|---|---|
| count(phrase in other documents) | count(all other phrases in all other documents) |

The independence score is calculated based on the observed ($O$) and expected ($E$) counts:

$$\chi^2 = \sum_{i,j} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

where $i, j$ are the row and column indices of the

contingency table. The $O$ counts are the cells of the table. The $E$ counts are calculated from the marginal probabilities (the sum of the values of a column or a row) converted into proportions by dividing them with the total number of observed events ($N$):

$$N = O_{1,1} + O_{1,2} + O_{2,1} + O_{2,2}$$

Then the expected count for seeing the phrase in the document is:

$$E_{1,1} = \frac{O_{1,1} + O_{1,2}}{N} \times \frac{O_{1,1} + O_{2,1}}{N} \times N$$

To measure the **phraseness** of a candidate phrase we use a technique based on the $\chi^2$ independence test. We measure the independence of the events of seeing the components of the phrase in the text. This method was found to be one of the best performing models in collocation discovery (Pecina and Schlesinger, 2006). For n-grams where $N > 2$ we apply the $\chi^2$ independence test by splitting the phrase in two (e.g. for a 4-gram, we measure the independence of the composing bigrams).

### 3.2 Discourse Comprehension Features

Very few existing keyword extraction methods look beyond word frequency. Except for (Turney and Littman, 2003), who uses pointwise mutual information to improve the coherence of the keyword set, we are not aware of any other work that attempts to use the semantics of the text to extract keywords. The fact that most systems rely heavily on term frequency properties poses serious difficulties, since many index entries appear only once in the document, and thus cannot be identified by features based solely on word counts. For instance, as many as 52% of the index entries in our training data set appeared only once in the books they belong to. Moreover, another aspect not typically covered by current keyword extraction methods is the coherence of the keyword set, which can also be addressed by discourse-based properties.

In this section, we propose a novel feature for keyword extraction inspired by work on discourse comprehension. We use a construction integration framework, which is the backbone used by many discourse comprehension theories.

#### 3.2.1 Discourse Comprehension

Discourse comprehension is a field in cognitive science focusing on the modeling of mental processes associated with reading and understanding text. The most widely accepted theory for discourse comprehension is the construction integration theory (Kintsch, 1998). According to this theory, the elementary units of comprehension are propositions, which are defined as instances of a predicate-argument schema. As an example, consider the sentence *The hemoglobin carries oxygen*, which generates the predicate CARRY[HEMOGLOBIN,OXIGEN]. The processing cycle of the construction integration model processes one proposition at a time, and builds a local representation of the text in the working memory, called the propositional network.

During the *construction* phase, propositions are extracted from a segment of the input text (typically a single sentence) using linguistic features. The propositional network is represented as a graph, with nodes consisting of propositions, and weighted edges representing the semantic relations between them. All the propositions generated from the input text are inserted into the graph, as well as all the propositions stored in the short term memory. The short term memory contains the propositions that compose the representation of the previous few sentences. The second phase of the construction step is the addition of past experiences (or background knowledge), which is stored in the long term memory. This is accomplished by adding new elements to the graph, usually consisting of the set of closely related propositions from the long term memory.

After processing a sentence, the *integration* step establishes the role of each proposition in the meaning representation of the current sentence, through a spreading activation applied on the propositions derived from the original sentence. Once the weights are stabilized, the set of propositions with the highest activation values give the mental representation of the processed sentence. The propositions with the highest activation values are added to the short term memory, the working memory is cleared and the process moves to the next sentence. Figure 3.2.1 shows the memory types used in the construction integration process and the main stages of the process.

#### 3.2.2 Keyword Extraction using Discourse Comprehension

The main purpose of the short term memory is to ensure the coherence of the meaning representation across sentences. By keeping the most important propositions in the short term memory, the spreading activation process transfers additional weight to se-

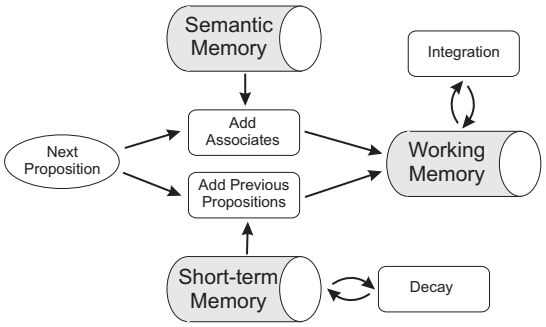


Figure 1: The construction integration process

mantically related propositions in the sentences that follow. This also represents a way of alleviating one of the main problems of statistical keyword extraction, namely the sole dependence on term frequency. Even if a phrase appears only once, the construction integration process ensures the presence of the phrase in the short term memory as long as it is relevant to the current topic, thus being a good indicator of the phrase importance.

The construction integration model is not directly applicable to keyword extraction due to a number of practical difficulties. The first implementation problem was the lack of a propositional parser. We solve this problem by using a shallow parser to extract noun phrase chunks from the original text (Munoz et al., 1999). Second, since spreading activation is a process difficult to control, with several parameters that require fine tuning, we use instead a different graph centrality measure, namely PageRank (Brin and Page, 1998).

Finally, to represent the relations inside the long term semantic memory, we use a variant of latent semantic analysis (LSA) (Landauer et al., 1998) as implemented in the InfoMap package,[2] trained on a corpus consisting of the British National Corpus, the English Wikipedia, and the books in our collection. To alleviate the data sparsity problem, we also use the pointwise mutual information (PMI) to calculate the relatedness of the phrases based on the book being processed.

The final system works by iterating the following steps: (1) Read the text sentence by sentence. For each new sentence, a graph is constructed, consisting of the noun phrase chunks extracted from the original text. This set of nodes is augmented with all the phrases from the short term memory. (2) A weighted edge is added between all the nodes, based on the semantic relatedness measured between the phrases by using LSA and PMI. We use a weighted combination of these two measures, with a weight of 0.9 assigned to LSA and 0.1 to PMI. For the nodes from the short term memory, we adjust the connection weights to account for memory decay, which is a function of the distance from the last occurrence. We implement decay by decreasing the weight of both the outgoing and the incoming edges by $n * \alpha$, where $n$ is the number of sentences since we last saw the phrase and $\alpha = 0.1$. (3) Apply PageRank on the resulting graph. (4) Select the 10 highest ranked phrases and place them in the short term memory. (5) Read the next sentence and go back to step (1).

Three different features are derived based on the construction integration model:

- **CI short term memory frequency** (*CI shortterm*), which measures the number of iterations that the phrase remains in the short term memory, which is seen as an indication of the phrase importance.

- **CI normalized short term memory frequency** (*CI normalized*), which is the same as *CI shortterm*, except that it is normalized by the frequency of the phrase. Through this normalization, we hope to enhance the effect of the semantic relatedness of the phrase to subsequent sentences.

- **CI maximum score** (*CI maxscore*), which measures the maximum centrality score the phrase achieves across the entire book. This can be thought of as a measure of the importance of the phrase in a smaller coherent segment of the document.

### 3.3 Syntactic Features

Previous work has pointed out the importance of syntactic features for supervised keyword extraction (Hulth, 2003). The construction integration model described before is already making use of syntactic patterns to some extent, through the use of a shallow parser to identify noun phrases. However, that approach does not cover patterns other than noun phrases. To address this limitation, we introduce a new feature that captures the part-of-speech of the words composing a candidate phrase.

[2] http://infomap.stanford.edu/

There are multiple ways to represent such a feature. The simplest is to create a string feature consisting of the concatenation of the part-of-speech tags. However, this representation imposes limitations on the machine learning algorithms that can be used, since many learning systems cannot handle string features. The second solution is to introduce a binary feature for each part-of-speech tag pattern found in the training and the test data sets. In our case this is again unacceptable, given the size of the documents we work with and the large number of syntactic patterns that can be extracted. Instead, we decided on a novel solution which, rather than using the part-of-speech pattern directly, determines the probability of a phrase with a certain tag pattern to be selected as a keyphrase. Formally:

$$P(pattern) = \frac{C(pattern,\ positive)}{C(pattern)}$$

where $C(pattern,\ positive)$ is the number of distinct phrases having the tag pattern $pattern$ and being selected as keyword, and $C(pattern)$ represents the number of distinct phrases having the tag pattern $pattern$. This probability is estimated based on the training collection, and is used as a numeric feature. We refer to this feature as *part-of-speech pattern*.

### 3.4 Encyclopedic Features

Recent work has suggested the use of domain knowledge to improve the accuracy of keyword extraction. This is typically done by consulting a vocabulary of plausible keyphrases, usually in the form of a list of subject headings or a domain specific thesaurus. The use of a vocabulary has the additional benefit of eliminating the extraction of incomplete phrases (e.g. "States of America"). In fact, (Medelyan and Witten, 2006) reported an 110% F-measure improvement in keyword extraction when using a domain-specific thesaurus.

In our case, since the books can cover several domains, the construction and use of domain-specific thesauruses is not plausible, as the advantage of such resources is offset by the time it usually takes to build them. Instead, we decided to use encyclopedic information, as a way to ensure high coverage in terms of domains and concepts.

We use Wikipedia, which is the largest and the fastest growing encyclopedia available today, and whose structure has the additional benefit of being particularly useful for the task of keyword extraction. Wikipedia includes a rich set of links that connect important phrases in an article to their corresponding articles. These links are added manually by the Wikipedia contributors, and follow the general guidelines of annotation provided by Wikipedia. The guidelines coincide with the goals of keyword extraction, and thus the Wikipedia articles and their link annotations can be treated as a vast keyword annotated corpus.

We make use of the Wikipedia annotations in two ways. First, if a phrase is used as the title of a Wikipedia article, or as the anchor text in a link, this is a good indicator that the given phrase is well formed. Second, we can also estimate the probability of a term $W$ to be selected as a keyword in a new document by counting the number of documents where the term was already selected as a keyword ($count(D_{key})$) divided by the total number of documents where the term appeared ($count(D_W)$). These counts are collected from the entire set of Wikipedia articles.

$$P(keyword|W) \approx \frac{count(D_{key})}{count(D_W)} \qquad (1)$$

This probability can be interpreted as "the more often a term was selected as a keyword among its total number of occurrences, the more likely it is that it will be selected again." In the following, we will refer to this feature as *Wikipedia keyphraseness*.

### 3.5 Other Features

In addition to the features described before, we add several other features frequently used in keyword extraction: the frequency of the phrase inside the book (*term frequency (tf)*); the number of documents that include the phrase (*document frequency (df)*); a combination of the two (*tf.idf*); the within-document frequency, which divides a book into ten equally-sized segments, and counts the number of segments that include the phrase (*within document frequency*); the length of the phrase (*length of phrase*); and finally a binary feature indicating whether the given phrase is a named entity, according to a simple heuristic based on word capitalization.

## 4 Experiments and Evaluation

We integrate the features described in the previous section in a machine learning framework. The system is evaluated on the data set described in Section 2.1, consisting of 289 books, randomly split into

90% training (259 books) and 10% test (30 books). We experiment with three learning algorithms, selected for the diversity of their learning strategy: multilayer perceptron, SVM, and decision trees. For all three algorithms, we use their implementation as available in the Weka package.

For evaluation, we use the standard information retrieval metrics: precision, recall, and F-measure. We use two different mechanisms for selecting the number of entries in the index. In the first evaluation (*ratio-based*), we use a fixed ratio of 0.45% from the number of words in the text; for instance, if a book has 100,000 words, the index will consist of 450 entries. This number was estimated based on previous observations regarding the typical size of a back-of-the-book index (Csomai and Mihalcea, 2006). In order to match the required number of entries, we sort all the candidates in reversed order of the confidence score assigned by the machine learning algorithm, and consequently select the top entries in this ranking. In the second evaluation (*decision-based*), we allow the machine learning algorithm to decide on the number of keywords to extract. Thus, in this evaluation, all the candidates labeled as keywords by the learning algorithm will be added to the index. Note that all the evaluations are run using a training data set with 10% undersampling of the negative examples, as described before.

Table 2 shows the results of the evaluation. As seen in the table, the multilayer perceptron and the decision tree provide the best results, for an overall average F-measure of 27%. Interestingly, the results obtained when the number of keywords is automatically selected by the learning method (decision-based) are comparable to those when the number of keywords is selected a-priori (ratio-based), indicating the ability of the machine learning algorithm to correctly identify the correct keywords.

Additionally, we also ran an experiment to determine the amount of training data required by the system. While the learning curve continues to grow with additional amounts of data, the steepest part of the curve is observed for up to 10% of the training data, which indicates that a relatively small amount of data (about 25 books) is enough to sustain the system.

It is worth noting that the task of creating back-of-the-book indexes is highly subjective. In order to put the performance figures in perspective, one should also look at the inter-annotator agreement between human indexers as an upper bound of performance. Although we are not aware of any comprehensive studies for inter-annotator agreement on back-of-the-book indexing, we can look at the consistency studies that have been carried out on the MEDLINE corpus (Funk and Reid, 1983), where an inter-annotator agreement of 48% was found on an indexing task using a domain-specific controlled vocabulary of subject headings.

### 4.1 Comparison with Other Systems

We compare the performance of our system with two other methods for keyword extraction. One is the *tf.idf* method, traditionally used in information retrieval as a mechanism to assign words in a text with a weight reflecting their importance. This *tf.idf* baseline system uses the same candidate extraction and filtering techniques as our supervised systems. The other baseline is the KEA keyword extraction system (Frank et al., 1999), a state-of-the-art algorithm for supervised keyword extraction. Very briefly, KEA is a supervised system that uses a Naïve Bayes learning algorithm and several features, including information theoretic features such as *tf.idf* and positional features reflecting the position of the words with respect to the beginning of the text. The KEA system was trained on the same training data set as used in our experiments.

Table 3 shows the performance obtained by these methods on the test data set. Since none of these methods have the ability to automatically determine the number of keywords to be extracted, the evaluation of these methods is done under the ratio-based setting, and thus for each method the top 0.45% ranked keywords are extracted.

| Algorithm | P | R | F |
|---|---|---|---|
| *tf.idf* | 8.09 | 8.63 | 8.35 |
| KEA | 11.18 | 11.48 | 11.32 |

Table 3: Baseline systems

### 4.2 Performance of Individual Features

We also carried out experiments to determine the role played by each feature, by using the information gain weight as assigned by the learning algorithm. Note that ablation studies are not appropriate in our case, since the features are not orthogonal (e.g., there is high redundancy between the construction integration and the informativeness features), and thus we cannot entirely eliminate a feature from the system.

| Algorithm | ratio-based | | | decision-based | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| Multilayer perceptron | 27.98 | 27.77 | 27.87 | 23.93 | 31.98 | 27.38 |
| Decision tree | 27.06 | 27.13 | 27.09 | 22.75 | 34.12 | 27.30 |
| SVM | 20.94 | 20.35 | 20.64 | 21.76 | 30.27 | 25.32 |

Table 2: Evaluation results

| Feature | Weight |
|---|---|
| part-of-speech pattern | 0.1935 |
| CI shortterm | 0.1744 |
| Wikipedia keyphraseness | 0.1731 |
| CI maxscore | 0.1689 |
| CI shortterm normalized | 0.1379 |
| ChiInformativeness | 0.1122 |
| document frequency (df) | 0.1031 |
| tf.idf | 0.0870 |
| ChiPhraseness | 0.0660 |
| length of phrase | 0.0416 |
| named entity heuristic | 0.0279 |
| within document frequency | 0.0227 |
| term frequency (tf) | 0.0209 |

Table 4: Information gain feature weight

Table 4 shows the weight associated with each feature. Perhaps not surprisingly, the features with the highest weight are the linguistically motivated features, including syntactic patterns and the construction integration features. The Wikipedia keyphraseness also has a high score. The smallest weights belong to the information theoretic features, including term and document frequency.

## 5 Related Work

With a few exceptions (Schutze, 1998; Csomai and Mihalcea, 2007), very little work has been carried out to date on methods for *automatic* back-of-the-book index construction.

The task that is closest to ours is perhaps keyword extraction, which targets the identification of the most important words or phrases inside a document. The state-of-the-art in keyword extraction is currently represented by supervised learning methods, where a system is trained to recognize keywords in a text, based on lexical and syntactic features. This approach was first suggested in (Turney, 1999), where parameterized heuristic rules are combined with a genetic algorithm into a system for keyphrase extraction (GenEx) that automatically identifies keywords in a document. A different learning algorithm was used in (Frank et al., 1999), where a Naive Bayes learning scheme is applied on the document collection, with improved results observed on the same data set as used in (Turney, 1999). Neither Turney nor Frank report on the recall of their systems, but only on precision: a 29.0% precision is achieved with GenEx (Turney, 1999) for five keyphrases extracted per document, and 18.3% precision achieved with Kea (Frank et al., 1999) for fifteen keyphrases per document. Finally, in recent work, (Hulth, 2003) proposes a system for keyword extraction from abstracts that uses supervised learning with lexical and syntactic features, which proved to improve significantly over previously published results.

## 6 Conclusions and Future Work

In this paper, we introduced a supervised method for back-of-the-book indexing which relies on a novel set of features, including features based on discourse comprehension, syntactic patterns, and information drawn from an online encyclopedia. According to an information gain measure of feature importance, the new features performed significantly better than the traditional frequency-based techniques, leading to a system with an F-measure of 27%. This represents an improvement of 140% with respect to a state-of-the-art supervised method for keyword extraction. Our system proved to be successful both in ranking the phrases in terms of their suitability as index entries, as well as in determining the optimal number of entries to be included in the index. Future work will focus on developing methodologies for computer-assisted back-of-the-book indexing, as well as on the use of the automatically extracted indexes in improving the browsing of digital libraries.

## Acknowledgments

We are grateful to Kirk Hastings from the California Digital Library for his help in obtaining the UC Press corpus. This research has been partially supported by a grant from Google Inc. and a grant from the Texas Advanced Research Program (#003594).

## References


S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7).

A. Csomai and R. Mihalcea. 2006. Creating a testbed for the evaluation of automatically generated back-of-the-book indexes. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*, pages 19–25, Mexico City.

A. Csomai and R. Mihalcea. 2007. Investigations in unsupervised back-of-the-book indexing. In *Proceedings of the Florida Artificial Intelligence Research Society*, Key West.

D. Feng, J. Kim, E. Shaw, and E. Hovy. 2006. Towards modeling threaded discussions through ontology-based analysis. In *Proceedings of National Conference on Artificial Intelligence*.

E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*.

M. E. Funk and C.A. Reid. 1983. Indexing consistency in medline. *Bulletin of the Medical Library Association*, 71(2).

A. Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, Japan, August.

W. Kintsch. 1998. *Comprehension: A paradigm for cognition*. Cambridge Uniersity Press.

M. Kubat and S. Matwin. 1997. Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the 14th International Conference on Machine Learning*.

T. K. Landauer, P. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25.

C. Manning and H. Schutze. 1999. *Foundations of Natural Language Processing*. MIT Press.

O. Medelyan and I. H. Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of the Joint Conference on Digital Libraries*.

M. Munoz, V. Punyakanok, D. Roth, and D. Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing*.

P. Pecina and P. Schlesinger. 2006. Combining association measures for collocation extraction. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 651–658, Sydney, Australia.

G. Salton and C. Buckley. 1997. Term weighting approaches in automatic text retrieval. In *Readings in Information Retrieval*. Morgan Kaufmann Publishers, San Francisco, CA.

H. Schutze. 1998. The hypertext concordance: a better back-of-the-book index. In *Proceedings of Computerm*, pages 101–104.

P. Turney and M. Littman. 2003. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems*, 4(21):315–346.

P. Turney. 1999. Learning to extract keyphrases from text. Technical report, National Research Council, Institute for Information Technology.

G. Weiss and F. Provost. 2001. The effect of class distribution on classifier learning. Technical Report ML-TR 43, Rutgers University.