Investigations in Unsupervised Back-of-the-Book Indexing

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

Abstract

This paper describes our experiments with unsupervised methods for back-of-the-book index construction. Through comparative evaluations performed on a gold standard data set of 29 books and their corresponding indexes, we draw conclusions as to what are the most accurate unsupervised methods for automatic index construction. We show that if the right sequence of methods and heuristics is used, the performance of an unsupervised back-of-the-book index construction system can be raised with up to 250% relative increase in F-measure as compared to the performance of a system based on the traditional *tf*idf* weighting scheme.

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 become available in electronic format, in projects such as Gutenberg (http://www.gutenberg.org), Google Book Search (http://books.google.com), or the Million Books project (http://www.archive.org/details/millionbooks). 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 concepts 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 however 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 investigate and evaluate a range of unsupervised methods for automatic back-of-the-book index construction. As it turns out, the traditional *tf*idf* term weighting scheme typically used in information retrieval and keyword extraction leads to poor results, which can be exceeded by a large margin by using a combination of heuristics, named entity recognition methods, information about the typical distribution of the phrases inside an index, and a differentiated view on the *informativeness* and the *phraseness* of the entries in the index.

Related Work

With a few exceptions, mainly concerned with finding the connection between index entries and their occurrence inside a book (Schutze 1998), to our knowledge no work has been published 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 parametrized heuristic rules are combined with a genetic algorithm into a system for keyphrase extraction (GenEx) that automatically identifies keywords in a document. Turney does not report on the recall of his system, only on the precision: a precision of 23.90% is obtained when five keyphrases are extracted per document, which drops to 12.80% when fifteen keyphrases are extracted. A different learning algorithm was used in the KEA system (Frank *et al.* 1999), where a Naive Bayes learning scheme is applied on the document collection, with results comparable to that of GenEx. Finally, in recent work, (Hulth 2003) applies a supervised

learning system to keyword extraction from abstracts, using a combination of lexical and syntactic features, proved to improve significantly over previously published results. Note that all this previous work has focused on the task of keyword extraction from short or medium-sized documents (reports), and no study to date has addressed the extraction of keyphrases from very large documents such as books.

Unsupervised Back-of-the-Book Indexing

The task of a back-of-the-book index construction system consists of automatically identifying the entries most likely to belong to a back-of-the-book index.

According to indexing guidelines established for professional indexers, the entries in an index typically consist of *subject headings* and *names*. Subject headings are expressions extracted from the text or generated by the indexer, which serve as pointers to key concepts described in the text (Knight 1979). The subject headings are usually domain specific, and consist of a simple noun, a phrase, or a combination of a phrase with some additional explanatory information generally increasing its specificity. Such an extended entry is referred to as a compound heading.

Another characteristic feature of the indexing style is the fact that the most important subphrase of a compound entry – the keyword (often termed as *head word* or *lead term*) – has to appear at the beginning of the entry. This results in entries where the order of the words is altered and the subparts are divided by commas; this is called an inverted index.

Figure 1 shows two snippets extracted from a back-of-the-book index. The first snippet contains several subject headings, as well as examples of inverted entries (*"illustrations, indexing of"*); the second snippet illustrates the high percentage of named entities.

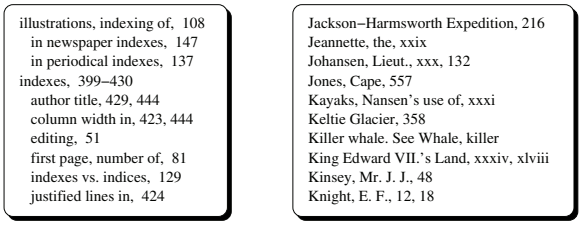

Figure 1: Examples of entries in a back-of-the-book index.

The traditional indexing literature advises the indexers to always select lead terms or complete headings that are present in the body of the text. "The keyword should, as a rule, be an actual word in the portion of the text being indexed, for it is awkward to consult an index and then not to find the word, given there as a keyword, in the part of the text indicated" (Knight 1979). This is an encouraging statement for us, since it indicates that extractive methods are plausible for automated back-of-the-book indexing. In fact, in our previous work (Csomai & Mihalcea 2006), we found that about 80% of the lead terms of the index were actually found in the text of the book.

It is worth noting that although the keywords are generally present in the text, this is not true for the other compound entries generated by the indexers. Consider for instance the entry *"indexes vs. indices,"* pointing the reader to the place in the text where the discussion about the proper plural form of the word index is found.[1] Such an entry will not be literally found in the text; therefore, given that we focus on *extractive* methods, such human generated entries will not be considered in the experiments reported in this paper.

Our system for unsupervised back-of-the-book index construction consists of three main steps. The system (1) employs a candidate extraction step that provides a set of likely index entries extracted from the original document, followed by (2) a ranking of the candidates that allows the system to choose the best candidates for the final index. Finally, (3) the quality of the system output is improved by using several filtering steps that eliminate unlikely candidates. In the following section, we describe each of these steps in detail.

Candidate Extraction Methods

The candidate extraction stage has a dual purpose: to extract the largest possible number of phrases likely to appear in the index, and at the same time minimize the number of incorrectly proposed candidates. We implemented two of the most frequently used keyword extraction methods previously proposed in the literature, and enhanced them with filtering heuristics and a module for named entity recognition, as described below.

The simplest and most widely used method is the *N-gram* approach, where all the possible sequences of N words in the text are extracted as candidates (throughout all our experiments, $N = \overline{1,4}$). This typically leads to an extremely large number of entries, which we filter by applying several restrictions meant to reduce the number of erroneous candidates, e.g. *N-grams* should not cross sentence boundaries, should not contain punctuation, etc.

The second method we consider is a *noun phrase chunks* approach, which has been used in the past for keyword extraction (Hulth 2003). The method relies on the empirical observation that the majority of the keyphrases consist of a noun phrase. This extraction method considerably reduces the size of the candidate set; the trade-off however is that some good candidates that are not noun phrases are now missed. To implement this method, we use the shallow parser available within the freely available SNoW architecture (Munoz *et al.* 1999). While one could argue that shallow parsing is a supervised method, since shallow parsing in itself is independent from the task of back-of-the-book indexing, and can be easily trained for a new language using relatively small amounts of data, we consider it applicable in our unsupervised framework.

One of our earlier observations concerning the composition of back-of-the-book indexes was that a considerable amount of the entries in an index consist of named entities (Csomai & Mihalcea 2006). This prompted our proposal for the third candidate extraction method that accounts for *named entities*. Similar to the noun phrase approach, this method yields a less noisy candidate set, but it also misses a

[1] In this context, the correct plural form is indexes.

significant number of potentially correct entries. In our experiments, this candidate extraction method was used with a dual role: in tandem with other extraction techniques, or as a ranking feature. To recognize named entities, we use a supervised approach consisting of the named entity recognition module incorporated in the LingPipe toolkit,[2] or, alternatively, we use an unsupervised heuristic that detects sequences of capitalized phrases that have not appeared before without capitalization. As illustrated in the evaluation section, this heuristic method leads to results comparable to those obtained with the supervised named entity recognition.

In addition to the implicit ability of some candidate extraction methods to eliminate sequences of words unlikely to appear in the index, we can define additional heuristics to further reduce the size of the candidate set. In the case of the *N-grams*, we use a heuristic that discards all the candidates that either begin or end with a stopword or a common word.[3] For the noun phrase chunks, instead of discarding the candidates, we remove the leading and trailing stopwords.

Ranking methods

The ranking stage is concerned with scoring and ordering the phrases obtained from the candidate extraction stage. Following (Tomokiyo & Hurst 2003), we distinguish between the *phraseness* and the *informativeness* of a keyphrase, and correspondingly devise methods to rank the candidate phrases according to each of these properties.

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. We attempt to extract the phrases that maximize both the phraseness and the informativeness properties, i.e. phrases that convey the most information to the reader and at the same time are perceived as complete coherent units.

To measure the **informativeness** of a keyphrase, we experimented with various methods previously proposed in the keyword extraction literature, namely:

- *tf*idf*, which is the traditional information retrieval metric (Salton & Buckley 1997), employed in most existing keyword extraction applications. We measure inverse document frequency using the British National Corpus (BNC), a large balanced corpus of modern English.
- *language model informativeness*, which was proposed in (Tomokiyo & Hurst 2003) as a way to discover keyphrases in a larger domain. While most of the other ranking methods were developed for shorter documents, the language model approach was specifically designed for large bodies of texts (collection of documents from a single domain), and therefore we also consider it in our evaluation. Under this model, the informativeness of a phrase is calculated as a combination of the probability of the phrase being observed in the book (foreground corpus) and in an unrelated collection (background corpus), using the *pointwise Kullback-Liebler divergence* measure of relative entropy (Tomokiyo & Hurst 2003).

$$inf_{lang}(phrase) = p_{fg}(phrase) \log \left(\frac{p_{fg}(phrase)}{p_{bg}(phrase)} \right)$$

- χ^2 *independence test*, which measures the degree to which two events happen together more often than by chance. In our case this translates into finding if a phrase occurs in the document more frequently than it would by chance. The information required for the χ^2 independence test can be typically summed up in a contingency table (Manning & Schütze 1999):

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

where e.g. *count(phrase in other documents)* stands for the number of times a given phrase appeared in a general corpus.

To measure the **phraseness** of a candidate phrase, we use techniques adopted from the collocation discovery literature.

- χ^2 *independence test*, which measures the independence of the events of seeing the components of the phrase in the text, and found to be one of the best performing models in collocation discovery (Pecina & Schlesinger 2006). For *N-grams* where $N > 2$ we apply the χ^2 independence test by splitting the phrase in two (e.g. for a 4-gram, we measure the independence of the composing bigrams).
- *language model phraseness*, which measures the *pointwise Kullback-Liebler divergence* between background and foreground probabilities; e.g. for bigrams of the form (a, b), we measure the pointwise divergence as:

$$phrase_{lang}(a, b) = p_{fg}(a, b) \log \left(\frac{p_{fg}(a, b)}{p_{bg}(a) p_{bg}(b)} \right)$$

For *N-grams* with $N > 2$, we split the *N-gram* in two, and calculate the divergence using the probability of occurrence of the smaller constituent *N-grams*.

Post-filtering

Our statistical approach does not consider the meaning of the extracted phrases, which leads to a number of undiserable effects, most notably the presence of paraphrases in the extracted set. To address this problem, we developed a simple paraphrase identification method that targets basic paraphrase types such as morpho-syntactic variations (*"law of nature"* and *"natural law"*) and lexical synonymy (*"nuclear weapons"* and *"atomic weapons"*).

The paraphrase recognition is carried out by replacing all the words in a candidate phrase with an extended set of tokens containing the word stems obtained using Porter's

[2] http://www.alias-i.com

[3] We use the stopwords (25) and common words (530) distributed with the Smart system ftp://ftp.cs.cornell.edu/pub/smart.

stemming algorithm (Porter 1980) and the synonyms defined in WordNet (Miller 1995), assuming the most frequent sense by default; stopwords are also discarded. Next, two such extended sets of tokens are matched if they share at least one token. Two candidate phrases are then labeled as paraphrases of each other if there is a bijective (one-to-one) mapping between the extended sets of each of the constituents words in the two candidates. Figure 2 shows a paraphrase identification example for the phrases *"employing the pendulum"* and *"pendulum applied."*

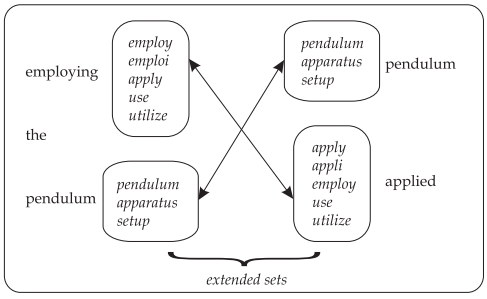

Figure 2: Example of paraphrase identification.

Combined models

A good keyphrase involves different components and aspects, which ideally should be captured by an automatic indexing system. We therefore propose several combination models where each candidate is scored based on a weighted linear combination of scores obtained with different models, where a model consists of a certain extraction or ranking method. Formally $s(c) = \sum_{m \in (E \times R)} \lambda_m score_m(c)$, where E and R are the set of extraction and ranking methods respectively, and λ_m is a weight used to control the importance of the model m in the combined score.

Evaluation

To evaluate the various back-of-the-book indexing methods, we needed a collection of books in electronic format, together with their manually generated back-of-the-book indexes. We use a subset of the collection we introduced in (Csomai & Mihalcea 2006). We use 29 of the original 55 electronic books in this collection, after removing the books dated from the early and mid 19-th century, which were observed to have indexes following an outdated, obsolete style, being composed of long, explicative sentences. We refer to this collection as the *gold standard index*, and our evaluations are based on comparisons against this data set.

We use the coarse index version, consisting of general concepts mentioned in the original index, and built by using the head expressions representing the most important part of the index entries. As we have previously shown (Csomai & Mihalcea 2006), this index version has the highest recall measured against the text of the book, which makes it appropriate for the evaluation of extractive methods that do not attempt to generate new entries, but mainly try to identify important sequences of words already present in the text.

Separate evaluations were conducted for the candidate extraction methods, filtering heuristics, and ranking methods, as described below.

Evaluating candidate extraction methods

To assess the performance of the candidate extraction methods, we determine the ability of these methods to extract the largest amount of candidates that actually appear in the gold standard index. This can be thought of as measuring the recall of a naive back-of-the-book indexing method consisting of only the candidate extraction, without any ranking or filtering. The *recall* on a single document can be measured as the number of candidates present in the gold standard, divided by the size of the gold standard. Secondly, we also need to measure the quality of the candidate set, that is how well it can filter out the candidates unlikely to be present in the gold standard index. This can be thought of as the *precision* of the naive approach, and can be quantified as the number of candidates present in the index divided by the number of candidates. In addition to precision and recall, we also determine the *F-measure*, calculated as the harmonic mean of precision and recall.

Method	P	R	F
N-grams	0.01	84.20	0.02
NP-chunks	4.40	28.99	7.64
NE	15.79	39.10	22.49
NEheur	11.43	43.34	18.08

Table 1: Micro precision (P), recall (R), and F-measure (F) for different candidate extraction methods. NE stands for supervised Named Entity recognition, and NEheur for the heuristic approach for Named Entity recognition.

Table 1 shows the precision, recall, and F-measure figures measured for the candidate extraction methods. Perhaps not surprisingly, the *N-gram* approach leads to the highest recall, at the cost of low precision. Looking at the total number of candidates generated by the *N-grams* method, more than 7 million unique phrases are extracted for the collection of 29 books, compared to only 90,000 phrases obtained with the *NP chunks* approach. In fact, the latter method has a significantly reduced recall, and a somehow higher precision. The performance of the *named entity recognition* method is even more remarkable: without any further ranking, its results are comparable to those obtained in keyphrase extraction. This demonstrates that our hypothesis that back-of-the-book indexes contain a large amount of named entities is correct. Interestingly, the named entity heuristic leads to results within a reasonable margin from the supervised named entity approach, which suggests that this unsupervised method is a valid alternative for those languages for which named entity recognition tools are not available.

Note that the recall values shown in Table 1 can be considered as an upper bound for the extractive methods, as no extractive system can achieve a recall higher than the one obtained by the candidate extraction stage.

	N-gram						NP chunk					
	no filtering			all filterings			no filtering			all filterings		
Method	P	R	F	P	R	F	P	R	F	P	R	F
*tf*idf*	10.33	11.12	10.71	15.76	16.48	16.11	10.86	11.02	10.94	16.13	16.24	16.19
χ^2	11.00	11.85	11.41	16.28	17.34	16.79	09.24	09.01	09.12	11.81	11.92	11.87
KL divergence	07.54	07.83	07.68	12.50	12.39	12.45	06.67	06.89	06.78	12.09	11.68	11.88

Table 3: Performance of ranking methods with various candidate extraction techniques

	N-gram						NP chunk					
	no filtering			all filterings			no filtering			all filterings		
Method	P	R	F	P	R	F	P	R	F	P	R	F
*tf*idf*	13.49	14.16	13.82	16.23	16.73	16.48	17.23	16.82	17.02	18.69	17.89	18.28
χ^2	17.06	17.76	17.40	19.50	19.99	19.74	17.23	16.60	16.91	18.04	17.29	17.65
KL divergence	08.71	09.33	09.01	11.75	11.90	11.83	11.30	11.47	11.38	12.26	11.90	12.07

Table 4: Performance of ranking methods with enforced distribution by length

Filtering	Precision	Recall	F-measure
none (baseline)	10.33	11.12	10.71
paraphrases	12.70	13.15	12.93
stopwords	14.25	15.08	14.66
common words	15.25	16.30	15.76
COMBINED	**15.77**	**16.48**	**16.11**

Table 2: Baseline system and improvements obtained with different filtering methods

Evaluating filtering methods

To evaluate the effect of the filtering heuristics (stopword and common word removal and paraphrase identification), we analyze their effect over a working system adopted as a baseline. The baseline system uses the *tf*idf* weighting scheme with an *N-gram* candidate extraction. The first row in Table 2 shows the results obtained for this baseline system. The subsequent rows show the scores of the system when different filtering methods are applied *individually* over the baseline system. The last row shows the performance of the system with all the filtering methods combined.

The evaluations show that the filtering can bring a 60% increase in F-measure over the baseline. Although not always with such a dramatical increase, a similar effect was observed in all the systems used in our evaluation. It is worth noting that the paraphrase filter has an additional, difficult to evaluate advantage, namely the fact that it eliminates duplicate candidates, which increases the perceived quality of the extracted index.

Evaluating ranking methods

Ranking methods are evaluated as an integral part of a working indexing system, along with the various candidate extraction methods. After the candidate phrases are extracted with one of the methods described in the previous sections, a ranking method is used to sort the phrases, and the top ranked phrases are selected for inclusion in the index.

Although the size of a back-of-the-book index varies with the domain, the academic level of the text, or the indexing style, the literature provides a general guideline indicating that the index should cover a number of pages of approximately 5% of the number of pages in the book (Knight 1979; Mulvany 2005). In our corpus, we observed an average index length of 0.35% of the total number of words in the text – which translates to a page-based percentage roughly equivalent to the guideline value. We decided to use and enforce this value across all the evaluations by extracting a corresponding amount of keyphrases for each book.

Table 3 shows the results of the various ranking methods with the *N-gram* and *noun phrase chunk* candidate extraction methods. We report the results both with and without the filtering methods, so that we can get a clear picture on the performance of the ranking methods and their full capability when improved with filtering techniques. The background corpus used for the χ^2 independence and KL-divergence metrics is the BNC corpus; for the KL-divergence informativeness we use a Good-Turing smoothing.

Contrary to our expectations, the best performance is obtained with a system using an *N-gram* candidate extraction. The *tf*idf* ranking performs about the same for both extraction methods. Surprisingly, the χ^2 performs much better with the *N-gram* extraction, where it outperforms the *tf*idf*. The KL divergence provided the worst results in both cases.

The length of a candidate phrase can also represent an important clue for keyphrase selection, since the average length of keyphrases varies across domains, and it is particularly sensitive to the indexing style (concise vs. long explicative phrases). In order to account for this property, we select the top candidates so that the distribution of the candidate lengths follows the distribution of the index entries in the gold standard. The results obtained using this constraint are shown in Table 4. As seen in the table, the distribution constraint brings an improvement for all the methods, in particular in the performance of the χ^2 ranking with the *N-gram* candidate selection, where the increase in F-measure is 6% in the unfiltered case, and about 3% in the filtered case. The consistent superiority of the *N-gram* extraction with χ^2 weighting leads us to the conclusion that for the task of back-of-the-book indexing this combination leads to the best keyphrase extraction and ranking.

In addition to the individual models, we have also evaluated several combination methods. In all the combination experiments, the *N-gram* extraction was used, sometimes coupled with named entity recognition. The named entity and named entity heuristic candidates are ranked using a $tf*idf$ weighting and have an empirically selected weight of $\lambda = 2$. All other models have an equal weight of $\lambda = 1$. All the filtering methods are used in the combination models. Table 5 shows the results obtained with several combination methods.

	P	R	F
tf*idf, χ^2Phrase	16.85	17.51	17.18
χ^2Inf, χ^2Phrase	20.65	21.25	20.95
KL-Inf, KL-Phrase	12.82	13.09	12.96
tf*idf, NE	**25.95**	**25.27**	**25.60**
χ^2Inf, χ^2Phrase, NE	**26.70**	**26.37**	**26.53**
KL-Inf, KL-Phrase, NE	22.63	22.36	22.49
tf*idf, NEheur	21.12	20.68	20.89
χ^2Inf, χ^2Phrase, NEheur	**23.17**	**23.14**	**23.15**
KL-Inf, KL-Phrase, NEheur	20.98	20.72	20.84
tf*idf baseline	10.33	11.12	10.71

Table 5: Performance of combined models; Inf=informativeness, Phrase=phraseness.

As shown in the table, all the combinations of phraseness and informativeness increased the performance compared to the informativeness models alone, which suggests that the initial assumption that phraseness is an important aspect of back-of-the-book index entries is correct. Another important observation is that all the combinations where named entities are included improve significantly over the other models, confirming our intuition that named entity recognition can improve the performance of an indexing system.

Conclusions

In this paper, we evaluated several unsupervised methods for back-of-the-book index construction, and showed that the right choice of methods and heuristics can lead to significant improvements compared to the *tf*idf* weighting scheme typically used for information retrieval and keyword extraction.

In particular, our experiments allowed us to draw the following conclusions: (1) keyword extraction and named entity recognition techniques can be successfully used for the purpose of back-of-the-book indexing, which means that the research literature available for these topics can be helpful in providing pointers for tasks relevant to back-of-the-book indexing; (2) it is useful to design the indexing system in stages, with different modules handling the candidate extraction, the phrase ranking, and the phrase filtering; (3) it is important to differentiate between the phraseness and the informativeness of a candidate phrase, which brings improvements over systems that do not make such a distinction; and finally (4) combination models can lead to results significantly better than those obtained with individual models.

In particular, the concrete lesson learned from our experiments is that the best performance of 26.53 F-measure can be obtained using a combined *N-gram* and named entity candidate extraction method with stopword, common word filters and paraphrase identification filters, coupled with a ranking scheme using χ^2 informativeness and χ^2 phraseness. In the absence of a named entity recognition system, a comparable 23.16 F-measure can be obtained using an unsupervised heuristic for named entity recognition. Overall, our unsupervised system leads to an increase in F-measure of up to 250% as compared to the baseline represented by a *tf*idf* weighting scheme, representing a significant improvement.

Acknowledgments

This work was partially supported by a research grant from Google Inc.

References

Csomai, A., and Mihalcea, R. 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*, 19–25.

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

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

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

Knight, G. N. 1979. *Indexing, The art of*. United Kingdom: The Gresham Press.

Manning, C. D., and Schütze, H. 1999. *Foundations of Statistical Natural Language Processing*. Cambridge, Massachusetts: The MIT Press.

Miller, G. 1995. Wordnet: A lexical database. *Communication of the ACM* 38(11):39–41.

Mulvany, N. C. 2005. *Indexing Books*. Chicago: The University of Chicago Press.

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

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

Porter, M. 1980. An algorithm for suffix stripping. *Program* 14(3):130–137.

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

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

Tomokiyo, T., and Hurst, M. 2003. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions*, 33–40.

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

Unsupervised learning (UL) is a type of algorithm that learns patterns from untagged data. The hope is that through mimicry, the machine is forced to build a compact internal representation of its world. In contrast to Supervised Learning (SL) where data is tagged by a human, eg. as "car" or "fish" etc, UL exhibits self-organization that captures patterns as neuronal predelections or probability densities. The other levels in the supervision spectrum are Reinforcement Learning where the machine is The output is the amount of time it took to drive back home on that specific day. You instinctively know that if it's raining outside, then it will take you longer to drive home. But the machine needs data and statistics. Let's see now how you can develop a supervised learning model of this example which help the user to determine the commute time. The first thing you requires to create is a training data set.Â You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labeled and not known. Summary. In Supervised learning, you train the machine using data which is well "labeled." Unsupervised learning is a machine learning technique, where you do not need to supervise the model. What's the difference between supervised, unsupervised, semi-supervised, and reinforcement learning? Based on the kind of data available and the research question at hand, a scientist will choose to train an algorithm using a specific learning model.Â SuperVize Me: Whatâ€™s the Difference Between Supervised, Unsupervised, Semi-Supervised and Reinforcement Learning? August 2, 2018 by Isha Salian. Share. Our system for unsupervised back-of-the-book index con-struction consists of three main steps. The system (1) em-ploys a candidate extraction step that provides a set of likely index entries extracted from the original document, followed by (2) a ranking of the candidates that allows the system to choose the best candidates for the nal index.Â 1999). While one could argue that shal-low parsing is a supervised method, since shallow parsing in itself is independent from the task of back-of-the-book in-dexing, and can be easily trained for a new language using relatively small amounts of data, we consider it applicable in our unsupervised framework. But what would the "target output" be for unsupervised learning? Can someone kindly provide an example of how you'd use BP in unsupervised learning, specifically for clustering of classification? Thanks in advance. machine-learning neural-network unsupervised-learning.Â Backpropagation is used to compute the derivatives of the error function for training an artificial neural network with respect to the weights in the network. It's named as such because the "errors" are "propagating" through the network "backwards". You need it in this case because the final error with respect to the target depends on a function of functions (of functions depending on how many layers in your ANN.)