Proceedings of the Twenty-Eighth International Florida Artificial Intelligence Research Society Conference

Keyphrase Extraction and Grouping Based on Association Rules

Xin Li and Fei Song
School of Computer Science, University of Guelph
Guelph, ON, Canada N1G 2W1
xli01@uoguelph.ca and fsong@uoguelph.ca

Abstract

Keyphrases are important in capturing the content of a document and thus useful for many natural language processing tasks such as Information Retrieval, Document Classification, and Text Summarization. Keyphrase extraction aims to identify multi-word sequences from a collection of documents that more or less correspond to keyphrases. In this paper, we propose a new method for keyphrase extraction based on association rule mining. Redundant multi-word sequences or synonymous phrases inevitably make up a big part of the keyphrases extracted. With association rules, we can also reduce the redundancy by grouping the related keyphrases that have strong co-occurrence frequencies. We further apply our keyphrase extraction and grouping solution to Information Retrieval. By both distinguishing and grouping keyphrases, we are able to achieve improved performance for Information Retrieval.

Introduction

A keyphrase is a sequence of words that usually follows a grammatical structure and describes a specific concept or entity. For example, "hot dog" and "Apple store" cannot be adequately represented by the individual words contained in them separately. Keyphrase extraction is to identify multi-word sequences that more or less correspond to keyphrases. Since keyphrases help capture the content of a document, they are useful for a wide range of natural language processing tasks such as Information Retrieval (Croft, Turtle, and Lewis 2010), Document Classification (Coenen et al. 2007), and Text Summarization (Wan and Xiao 2010).

Although keyphrases can be manually identified by humans like what is commonly done by librarians for indexing books and other references, it is, however, not feasible to do the same for large document collections that are increasingly available with the fast growth of the Internet and the Web. To avoid such labor-intensive and time-consuming tasks, it is strongly desirable to automate the extraction of keyphrases so that we can have a scalable technique to keep up with the fast growth of the document collections.

In addition to the extraction of keyphrases, there is also a need to group them since some may have similar meanings. For example, the full name of "Twentieth Century Fox" is "Twentieth Century Fox Film Corporation". For the NBA player Michael Jordan, his full name is Michael Jeffrey Jordan and sometimes he is just called Jordan. Finding ways to group these keywords and keyphrases can help us not only cut down the redundancy in text representation but also find as many relevant documents as possible for applications like Information Retrieval.

In this paper, we focus on the automatic extraction and grouping of keyphrases along with keywords. Our main contribution is to apply association rule mining for both tasks so that they can be handled consistently in one framework. We first identify frequent word sequences using the BIDE algorithm (Wang and Han 2004), which are then strengthened with a LocalMaxs method (Silva et al. 1999). After that, we group synonymous keyphrases with association rules based on co-occurrence frequencies. We further apply our solution for keyphrase extraction and grouping to Information Retrieval so that we can evaluate its performance on a standard dataset.

For the rest of the paper, we first introduce the related work before presenting our new solutions to keyphrase extraction and grouping. After that, we describe our experimental results along with discussions. Finally, we conclude our paper with a discussion on future directions.

Related Work

Keyphrase Extraction and Grouping

Keyphrase extraction has been studied for a long time and we can broadly divide the current research into linguistic, statistical, graph-based, and hybrid approaches. Linguistic approaches use grammatical structures at morphological, syntactic, and/or semantic levels to extract phrases. Statistical approaches use different association measures to identify frequent phrases from a corpus. Graph-based approaches use tokens and the co-occurrence relations and/or semantic relations among them to construct a graph for finding the most likely phrases. Hybrid approaches are combinations of the approaches mentioned above.

The advantage of linguistic approaches is that they can generate less noisy output, but the disadvantage is that they are not easily adaptable to a new domain, since the rules are usually constructed manually for a specific domain.

Statistical approaches, on the other hand, can be noisy, but are usually domain-independent and scalable for large datasets, since the process can often be automated through machine learning techniques. Various association measures have been proposed such as those based on term frequencies, TF×IDF, co-occurrences, and term independence so that keyphrases can be extracted by identifying word sequences that are strongly associated according to such measures.

One particular statistical method, called LocalMaxs, is proposed by Silva et al. (1999) that uses the lexical connection to extract keyphrases. The authors describes the lexical connection as "glue" values, which are usually strong within true phrases, such as "Twentieth Century Fox Film Corporation" but weak within arbitrary multi-word sequences such as "if his cats" and "I will be". Based on this intuition, they propose their LocalMaxs algorithm as follows. Assume W is an n-gram $w_1 w_2 ... w_n$, Y is a (n+1)-gram extended from W by adding another word before, within, or after W, and X is a (n-1)-gram reduced from W by removing a word. Then, W will be chosen as a keyphrase if:

$$length(W) = 2 : g(W) > g(Y)$$

$$length(W) > 2 : g(X) \leq g(W) \text{ and } g(W) > g(Y)$$

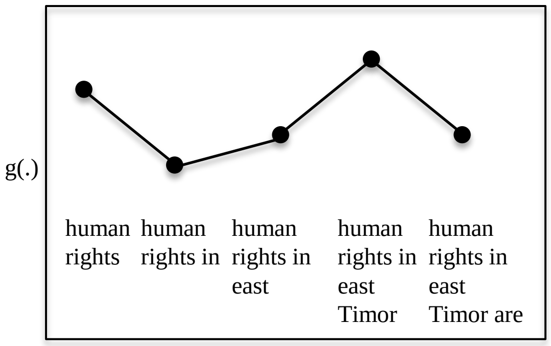

Figure 1: Example "Glue" Values for Some N-grams

Figure 1 shows an example from Silva et al. (1999) that clearly illustrates the idea of LocalMaxs. As can be seen, the glue values for "Human Rights" and "Human Rights in East Timor" are locally higher than their (n-1)-grams and (n+1)-grams. As a result, they will be extracted as keyphrases. The advantage of LocalMaxs is that it needs neither complex linguistic information nor empirically obtained thresholds for keyphrase extraction. In addition, it is a pure statistical approach that can be easily applied to different natural languages.

By grouping similar phrases into clusters, we can increase the recall performance of a text processing system. Existing work is usually based on the fact that similar documents tend to share many phrases in common. As a result, we can use the co-occurrence patterns to group related phrases together. Zamir (1999) tries to cluster documents based on the phrase clusters. Latiri (2012) applies phrase grouping to query expansion, where a concept iceberg lattice is used to find the maximal phrase groups that share the same documents. Kuhn et al. (2010) uses phrase clustering for statistical machine translation, where phrases are grouped according to two types of similarity metrics: count-based and edit-based. The former relies on the co-occurrence counts of phrases, such as "law" and "court", and the latter focuses on the makeup of the phrases, such as "International Business Machine" and "Business Machine". They got a better result for the count-based clustering, but also found that the result can be unreliable when the training data is too small.

Association Rule Mining

Association rules can capture important relations between variables in large databases, and mining association rules is useful for a wide range of applications. Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items and $D = \{d_1, d_2, \ldots, d_m\}$ be a database of transactions where each $d_i \subseteq I$. Given two itemsets: $A \subset I$, $B \subset I$, and $A \cap B = \emptyset$, an association rule of the form $A \Rightarrow B$ holds if it has a strong support ($\geq min\_sup$) and a strong confidence ($\geq min\_conf$) where support($A \Rightarrow B$) equals to the probability of P($A \cup B$) and confidence($A \Rightarrow B$) equals to P($A|B$) with respect to the transactions in D.

To generate association rules, we normally need to identify all frequent itemsets (i.e., those with support values $\geq min\_sup$). One popular solution is the Apriori algorithm proposed by Agrawal and Srikant (1994), which first identifies frequent 1-itemsets in a database and extends them with a single item to get 2-itemsets. After that, it prunes those 2-itemsets that do not satisfy the min_sup value. Such a process is repeated level-by-level until no frequent k-itemsets can be found. The algorithm is costly in both memory and time in that it needs to generate and maintain a large number of candidate itemsets and repeatedly scan the database for counting their frequencies.

To extract keyphrases, we actually need frequent sequences out of individual documents in a text collection so that the word order can be represented and distinguished. Fortunately, there is an efficient algorithm called BIDE (BI-Directional Extension) (Wang and Han 2004) that allows us to find all frequent sequences without the need to maintain the large number of candidates along with two pruning procedures to optimize the search process. In fact, BIDE only generates frequent closed sequences in order to reduce the redundant frequent sequences. A frequent sequence is closed if its support value $\geq min\_sup$ and no sequences that contain it as a subsequence have the same support values. As a result, we can cut down the redundancy of the shorter sequences that have the same support values of the related frequent closed sequences.

Proposed Solution

In this section, we describe our proposed solution for keyphrase extraction and grouping along with its application to information retrieval.

Generating Keyphrase Candidates

The BIDE algorithm allows us to generate sequences with gaps. For example, given the string “ABCDEFG”, a sequence can be “ACG” where there are gaps between A and C and between C and G in the original string. To identify keyphrases, we may need to skip some words in text, such as extracting “pick up” from “pick it up”. However, most keyphrases are noun phrases, for which we rarely skip words in between. If the stop words like “it”, “of”, and “the” are removed during text preprocessing, as is normally done for many language processing tasks, even the case like “pick up” will not need to skip any words. As a result, we simply adapt the BIDE algorithm to extract only consecutive words or ngrams as the sequences. This also helps improve the efficiency since there is no need to skip any words in the search process.

In addition, although frequent closed sequences help reduce the redundancy in text representation, we still need to extract many frequent sequences for applications like information retrieval. This is because a user may use different frequent keyphrases in queries and if we only store frequent closed sequences in the index, we will not be able to match them with some of the frequent sequences in queries, resulting in low recall performance. Consequently, we also need to extract frequent sequences along with the frequent closed sequences using a BIDE-like algorithm. One main advantage of such an algorithm is that it saves both memory and time in generating frequent sequences since all sequences to be examined are maintained in pseudo-projection databases in the form of positions and offsets of the sequences in the original database.

Frequent sequences are selected based on document frequencies, measuring how often they occur across all documents in a database. They are sometimes not strong enough to be keyphrases. For example, “able to find”, “fast enough”, and “where is it” are frequent sequences, but they are not normally considered as keyphrases. As a result, we need to further differentiate frequent sequences by selecting strong ones as keyphrases, as discussed in the following subsection.

Selecting Keyphrases

From frequent sequences, we further apply the LocalMaxs method so that sequences with strong word associations are selected as keyphrases. The method uses a “glue” value to measure the strength of word associations. It starts with the Symmetric Conditional Probability (SCP) between two words x and y (Silva et al. 1999):

$$SCP(x, y) = p(x|y)p(y|x) = \frac{p(x, y)^2}{p(x)p(y)} \tag{1}$$

where $p(x|y)$ is the conditional probability of word x occurring when word y appears, and $p(x, y)$ is the joint probability of words x and y occurring together.

To generalize the SCP to multi-word sequences, Fair Dispersion Point Normalization is defined as the average strength of splitting an n-gram at different points:

$$Avp = \frac{1}{n-1} \sum_{i=1}^{n-1} p(w_1, ..., w_i)p(w_{i+1}, ..., w_n) \tag{2}$$

Using Avp, we can then get the generalized SCP for sequences with $n \geq 2$ words:

$$SCP\_f(w_1, w_2, ..., w_n) = p(w_1, w_2, ..., w_n)^2/Avp \tag{3}$$

In Huo (2012), another association measure is proposed which normalizes the sequence probability by its nth root to get the average word-level strength for a sequence:

$$seq\_p(w_1, w_2, ..., w_n) = \sqrt[n]{p(w_1, w_2, ..., w_n)} \tag{4}$$

Both SCP_f and seq_p can be used as the glue values g(.) for the LocalMaxs method described the ”Related Work” section.

Grouping Related Keyphrases

With the keyphrases selected by LocalMaxs, we can then use them to group related frequent sequences based on an Iceberg lattice that uses confidence values or co-occurrence frequencies.

Grouping Based on Confidence Values An Iceberg lattice provides an intuitive visualization in the form of a hierarchical representation that shows the “includes” relationship among frequent sequences. As stated earlier, LocalMaxs helps us to select strong frequent sequences as keyphrases. However, some eliminated sequences may still appear in documents and queries, and if we do not include them in the index, we may fail to match them for information retrieval. By grouping these frequent sequences with the keyphrases selected via LocalMaxs, we can treat those sequences in each group as “synonyms” and thus increase the recall performance for information retrieval.

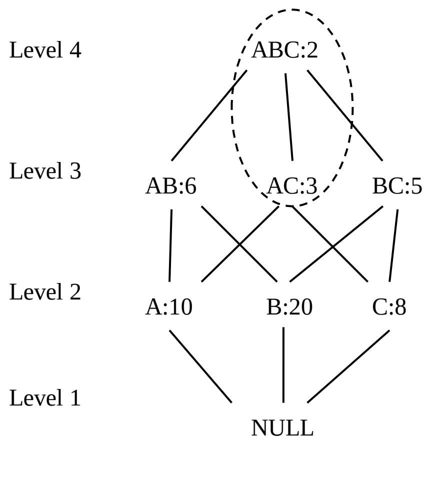

Figure 2: An Example Iceberg Lattice

An iceberg lattice is built from bottom up, but the search for synonym groups is done from top down. As illustrated in

Figure 2, level 1 links NULL (the empty sequence) to all unigrams; level 2 links all unigrams to bigrams, and so on. The top-down search looks for sequences that satisfy a minimum confidence in order to form synonym groups. Also in Figure 2, we assume that sequences in boldface are selected by LocalMaxs, which also includes all the unigrams for completeness. Using the minimum confidence of 0.6, we will start the search from the top node of ABC. We first search its sub-sequences and check their confidence values relative to ABC. Since AC is the only sequence that satisfies this minimum confidence, it is put into the same group with ABC. Then, AC is checked against its sub-sequences to see if more sub-sequences can satisfy the minimum confidence, and so on.

Grouping Based on Co-occurrence Frequencies Alternatively, we can group frequent sequences based on their co-occurrence frequencies in the database. In TextRank (Mihalcea and Tarau 2004), the co-occurrence frequency is computed within a window of maximum N words in a sentence, where N is a user-defined parameter between 2 and 10 in their experiments. To use co-occurrence frequencies to group frequent sequences, we compute the co-occurrence frequency of two sequences at a document level and propose a more relaxed condition for merging sequences: any two sequences that co-occur in a sufficient number of documents will be merged into one synonym group. More specifically, given $F(A)$ as the document frequency of sequence A, $U(A,B)$ as the co-occurrence frequency of sequences A and B, and R as the user-defined ratio for grouping, A and B can be put into one group if:

$$\frac{min(F(A), F(B))}{F(A) + F(B) - U(A,B)} \geq R \tag{5}$$

In Figure 3, we construct an example lattice using the LATIMES dataset. Each node contains a frequent sequence along with its document frequency. The nodes with circles correspond to the keyphrases selected by LocalMaxs. When we search the lattice from top down, "blood's ability carry" will be added to the group with "blood's ability carry oxygen" based on the above formula, but "blood's ability" will not be added since it does not meet the requirement above.

The advantage of co-occurrence frequencies is that by having a high value between two sequences, they become the "recommenders" to each other, thus helping us to connect them together to improve both recall and precision for an information retrieval system.

Application to Information Retrieval

To demonstrate the effectiveness of our keyphrase extraction and grouping method, we apply it to the task of information retrieval. We follow the typical process of creating the keyword index, including stopword removal, stemming, and document frequency cutoff for the rare words. However, instead of relying only on keywords, we use both keywords and keyphrases in our index for an information retrieval system.

With both keywords and keyphrases in the index, the challenge is to match queries against the index to identify

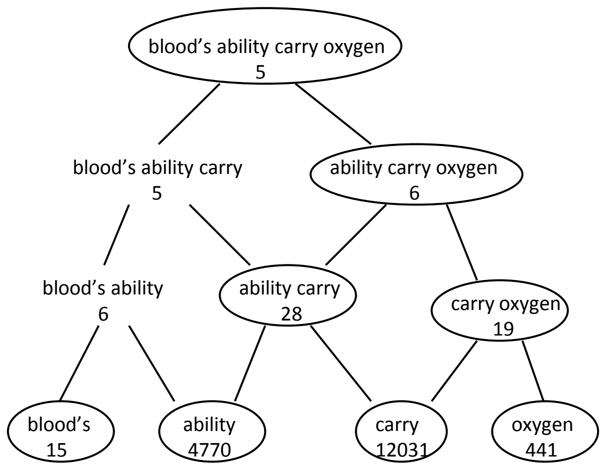

Figure 3: An Example Co-occurrence Graph

relevant documents. Using only keywords, the matching is straightforward: two words are either the same or different. For keyphrases, however, the matching is a bit more complicated: a piece of text can be matched only once or multiple times, and for a matched sequence, we can use only the sequence itself or also its subsequences, allowing overlaps of the embedded words. This leads to a total of four different variations as summarized in the following: (1) greedy match without overlaps (GN); (2) greedy match with overlaps (GO); (3) incremental match without overlaps (IN); and (4) incremental match with overlaps (IO). By greedy match, we try to search forward as far as possible for a sequence that can match a keyphrase in the index. After that, we skip beyond this sequence to search for the next match. For example, given the string "ABCDEFGHI", if "ABCD" is a longest match, we will move to "E" in the string to search for the next longest match. By incremental match, we only skip forward by one word after a longest matched sequence and then start searching for the next longest matched sequence. For the example above, after the longest match of "ABCD", we will move to "B" in the string to search for the next longest match. Without overlaps, only the longest matched sequence is used for retrieval. With overlaps, all the sub-sequences of the longest matched sequence are also used for retrieval if they are matched against certain keyphrases in the index as well.

To incorporate keyphrase grouping into an information retrieval system, we need to select a representative keyphrase for each synonym group so that a hashmap can be used to map all keyphrases to their corresponding representative keyphrases. To help identify a unique representative for each group, we first search for the longest keyphrases, and if there are more than one of them, we try to find the one that is located the earliest in the alphabetical order. After that, we merge all statistics (such as TF's and IDF's) for the keyphrases of the same groups and store them with the related representative keyphrases in the index. During retrieval, all matched keyphrases will be mapped to their representative keyphrases so that the merged statistics can be used for computing TF×IDF weights and cosine similarities. Thus, keyphrase grouping essentially allows us to connect more keyphrases together so that we can potentially enhance the recall performance of the related information retrieval system.

Experimental Results and Discussions

By including both keywords and keyphrases in the index, we can evaluate the impact of keyphrase extraction and grouping for information retrieval using the common measures on the standard datasets.

Datasets and Evaluation Metrics

Our keyphrase grouping algorithm has the time complexity of O(n^2). Due to the time-consuming process, we choose only the LATIMES dataset from TREC5 document collections for our experiments. The dataset has 131,896 documents, with an average of 526.5 words per document. For evaluations, we use the query sets from TREC 6, 7, and 8 in the form of 3-fold cross validation: i.e., two query sets for training and one for testing, and we repeat the process three times for the three folds.

Following the recent practice for ad hoc information retrieval, we use the Mean Average Precision (MAP) as the composite measure for evaluating the performance of an information retrieval system. It sums each query's Average Precision (AP) and divides the total by the number of queries. More specifically, MAP is defined as follows:

$$MAP = 1/N \sum_{j=1}^{N} (1/Q_j \sum_{i=1}^{Q_j} P(rel = i)) \qquad (6)$$

where Q_j is the number of retrieved relevant documents for query j; N is the number of queries; and $P(rel = i)$ is the precision at the ith relevant document.

Baseline for Information Retrieval

For all of our information retrieval systems, we follow the vector space model and use TF×IDF weights and cosine similarities to rank the retrieved documents for a query. For the baseline system, we only use frequent keywords (i.e., no keyphrases) in the index. Through the three-fold cross validation, we find that a threshold of 4 gives the best MAP result of 0.1256. As a result, we only keep the keywords that occur in four or more documents as frequent keywords in all of our information retrieval systems.

Keyphrase Matching Based on LocalMaxs

As described earlier, there are two glue functions (SCP_f and seq_p) that can be used by the LocalMaxs algorithm to select strong keyphrases. By including keyphrases in the index and using one of the four keyphrases matching methods described in the previous section, we get four different information retrieval systems along with their MAP results in Table 1.

Except for GN, the other three matching methods all did better than the baseline. The IN method, corresponding to "Incremental Match without Overlaps", did the best with the MAP values of 0.1532 for the SCP_f measure and 0.1535 for the seq_p measure, both of which are significantly better than the baseline at 95% confidence level. In terms of the number of keyphrases selected, the LocalMaxs based on seq_p has more than that based on SCP_f: 4,577,454 vs. 3,550,612, respectively, implying the SCP_f measure tends to be more selective in choosing keyphrases.

Table 1: Keyphrase Selection Based on LocalMaxs

Matching	LocalMax(SCP_f)	LocalMaxs(seq_p)
GN	0.1046	0.0932
GO	0.1498	0.1506
IN	**0.1532**	**0.1535**
IO	0.1501	0.1492

Keyphrase Matching Based on Lattice Grouping

To group frequent sequences with the related keyphrases selected by LocalMaxs, we set the confidence value to 0.95 for our lattice-based grouping method.

Table 2: Results Based on Lattice Grouping

Matching	LocalMax(SCP_f)	LocalMaxs(seq_p)
GN	0.1018	0.0919
IN	**0.1524**	0.1522

As can be seen in Table 2, the performance is decreased slightly for both GN and IN matching methods. By examining the individual results, we find that the ranking of the retrieved documents gets worse for some queries, indicating that the lattice grouping may introduce noise in merging frequent sequences into groups. Note that the overlapped matching methods are not suitable for keyphrase grouping, since the short sequences will be over-counted in the process.

Keyphrase Matching Based on Co-occurrence Frequencies

In addition to the lattice-based grouping, we can add the grouping based on co-occurrence frequencies. We also set the confidence level to 0.95, which is comparatively strong. As shown in Table 3, the performance for IN is improved significantly over that for IN using LocalMaxs alone. Thus, keyphrase extraction and grouping together is more effective than the other systems we developed for information retrieval.

Table 3: Results Based on Co-occurrence Grouping

Matching	LocalMax(SCP_f)	LocalMaxs(seq_p)
GN	0.1093	0.0983
IN	**0.1586**	0.1547

Encouraged by the results above, we try to decrease the confidence value, but unfortunately, the performance starts to decrease as well, indicating that weak confidence levels are not suitable for information retrieval.

Conclusions and Future Work

In this paper, we focused on the effective ways for keyphrase extraction and grouping based on association rule mining along with its application to information retrieval. For keyphrase extraction, we first adapt the BIDE algorithm for generating frequent ngrams which dramatically reduces the memory space by avoiding enumerating all possible ngrams and at the same time speeds up the process by pruning the search space.

To extract meaningful keyphrases from the frequent ngrams, we further apply the LocalMaxs method that does not require linguistic knowledge and is also language independent. However, although LocalMaxs helps extract high quality keyphrases, it is a bit too aggressive in that some useful keyphrases may be filtered out, making it difficult to match the related keyphrases for information retrieval. Consequently, we explore the use of lattice structures and co-occurrence frequencies for keyphrase grouping so that the frequent ngrams generated by BIDE can be merged with the keyphrases extracted by LocalMaxs to form synonym groups. All keyphrases in a related group can be matched against each other for information retrieval.

To demonstrate the effectiveness of our solution for keyphrase extraction and grouping, we apply it to information retrieval. With keyphrases, there can be different ways to match them in documents and queries. We tested our implementations on the standard TREC datasets. Our results indicate that adding frequent ngrams improves the retrieval performance significantly over the baseline made of keywords only. In addition, the performance can be further improved by selecting high quality keyphrases with the LocalMaxs method. Finally, by merging related keywords and keyphrases into synonym groups, we can increase the MAP value to 0.1586 from the baseline result of 0.1256, demonstrating the benefits of keyphrase extraction and grouping for information retrieval.

For keyphrase extraction, it would be useful to integrate linguistic knowledge, such as Part-of-Speech (POS) tags during data preprocessing and test on our datasets to see if the performance can be further improved.

In our experiments, we use a simple mechanism for keyphrase grouping which relies on the co-occurrence frequencies of keyphrases. For future work, we could explore other semantic relations among keyphrases to enhance the quality of keyphrase grouping, such as those used in PageRank, Wikipedia, and WordNet. In particular, we could bring Wikipedia's semantic relations into groups right after the LocalMaxs is applied to keyphrase extraction.

In our experiments, we used the vector space model for information retrieval. It would be interesting to apply keyphrase extraction and grouping to other retrieval models such as language models. Due to the time consuming process of our solution for keyphrase grouping, we only tested our solutions on a subset of the whole TREC4&5 datasets. With further improvements on the efficiency of our keyphrase grouping algorithm, we could test our system on the whole TREC4&5 datasets to see if our results can be scaled up for larger datasets.

References

Agrawal, R., and Srikant, R. 1994. Fast algorithms for mining association rules. In *Proceedings of the 20th Conference on Very Large Data Bases*, 387–499.

Coenen, F.; Leng, P.; Sanderson, R.; and Wang, Y. J. 2007. Statistical identification of key phrases for text classification. *Machine Learning and Data Mining in Pattern Recognition* 4571:838–853.

Croft, W. B.; Turtle, H. R.; and Lewis, D. D. 2010. The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th SIGIR Conference*, 32–45.

Huo, W. 2012. Automatic multi-word term extraction and its application to web-page summerization.

Kuhn, R.; Chen, B.; Foster, G.; and Stratford, E. 2010. Phrase clustering for smoothing tm probabilities or how to extract paraphrases from phrase tables. In *Proceedings of the 23rd Internationall Conference on Computational Linguistics*, 608–616.

Latiri, C. C.; Haddad, H.; and Hamrouni, T. 2012. Towards an effective automatic query expansion pprocess using an association rule mining approach. *Journal of Intelligent Information Systems* 39(1):209–247.

Mihalcea, R., and Tarau, P. 2004. Textrank: Bringing order into texts. In *Proceedings of EMNLP*, 404–411.

Silva, J. F. D.; Dias, G.; Guillore, S.; and Lopes, J. G. P. 1999. Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units. *Lecture Notes in Computer Science* 1695:113–132.

Wan, X., and Xiao, J. 2010. Exploiting neighborhood knowledge for single document summarization and keyphrase extraction. *ACM Trans. on Information Systems* 28(2).

Wang, J., and Han, J. 2004. Bide: Efficient mining of frequent closed sequences. In *IEEE Proceedings of 20th International Conference on Data Engineering*, 79–90.

Zamir, O. E. 1999. *Clustering Web Documents: a Phrase-Based Method for Grouping Search Engine Results*. Ph.D. Dissertation, University of Washington.