SGRank: Combining Statistical and Graphical Methods to Improve the State of the Art in Unsupervised Keyphrase Extraction

Soheil Danesh
University of Colorado
Boulder
soheildb@gmail.com

Tamara Sumner
University of Colorado
Boulder
tamara.sumner@
colorado.edu

James H. Martin
University of Colorado
Boulder
james.martin@
colorado.edu

Abstract

Keyphrase extraction is a fundamental technique in natural language processing. It enables documents to be mapped to a concise set of phrases that can be used for indexing, clustering, ontology building, auto-tagging and other information organization schemes. Two major families of unsupervised keyphrase extraction algorithms may be characterized as statistical and graph-based. We present a hybrid statistical-graphical algorithm that capitalizes on the heuristics of both families of algorithms and is able to outperform the state of the art in unsupervised keyphrase extraction on several datasets.

1 Introduction

Keyphrase extraction algorithms aim to extract, from within the document phrases and words that best represent the document's main topics. Being able to accurately determine what a document is about allows computers to cluster together documents that share topics (Hammouda et al., 2005), better answer search queries (Qiu et al., 2012), and generate short document summaries (D'Avanzo et al., 2004). Furthermore, keyphrase extraction can be used to facilitate the automatic construction of concept maps (Leake et al., 2003) or ontologies (Fortuna et al., 2006) which enable better understanding of the interconnections and relations between different topics. Keyphrase extraction is also used in content-based recommender systems which help users in discovering information relevant to their previously expressed interests (Lops et al., 2011). The aforementioned techniques are all important tools in the organization and understanding of the ever expanding repositories of textual information available online in the form of research papers, news articles, blog posts, etc. and keyphrase extraction is central to all of them. Therefore it could be said that keyphrase extraction is a fundamental NLP task, improvements in which could cascade into improvements in higher-level applications that build upon it.

In this work we have focused on unsupervised keyphrase extraction approaches as not only they are useful in domains where training data is hard to procure but even in the presence of ample training data word weights calculated using unsupervised methods can be used as one of several features in supervised keyphrase extraction algorithms. Therefore increases in the accuracy of unsupervised methods can propagate into the results of supervised algorithms as well.

There are two prominent families of unsupervised keyphrase extraction algorithms. The older of these two is clustered around the tf-idf term weighting metric where word statistics such as frequency of occurrence in the document or rareness in the corpus are used to distinguish potential keyphrases. The more recently developed of the two families has been built on the foundation of the TextRank algorithm (Mihalcea & Tarau, 2004). In algorithms of this family a graphical representation of the text is constructed with words as nodes and edges reflecting co-occurrence relations. This graph is then used to run node ranking algorithms such as PageRank (Page

*Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*, pages 117–126,
Denver, Colorado, June 4–5, 2015.

et al., 1999) that assign weights to the node-words reflecting their semantic importance to the text.

Although some overlap between these two families of algorithms has occurred in works that incorporate statistical heuristics into graph-based methods this overlap is small and most methods do not utilize the full set of statistical heuristics. Our aim has been to 1) Construct a keyphrase extraction algorithm based on optimal statistical features and 2) Combine it with a graph-based algorithm for further improvements. The advantage of graph-based methods is that they take into account term co-occurrence patterns that are not generally utilized by statistical methods which take a bag of n-grams approach to document representation.

2 Related Works

In this section we focus mainly on related unsupervised keyphrase extraction algorithms. One of the most prominent of these algorithms has been the term frequency-inverse document frequency (tf-idf) term weighting function (Salton et al, 1975). Given a corpus of documents the tf-idf weight of term t in document d is mathematically expressed as $tf\text{-}idf(t,d)=tf(t,d)*idf(t)$ where $tf(t,d)$ is the frequency of term t in document d and $idf(t)=log(N/df(t))$ where N is the total number of documents in the corpus and $df(t)$ is the number of documents in the corpus that contain term t (Jones, 1972). The term frequency heuristic is based on the intuition that terms which occur more often in a document are more likely to be important to its meaning. The idf function captures the rareness heuristic, that is, words which occur in many documents in the corpus are unlikely to be important to the meaning of any specific one.

Tf-idf is simple yet relatively accurate therefore many variations of it have been used by other algorithms. One of the most successful of these is KP-Miner (El-Beltagy & Rafea , 2010) which to the best of our knowledge represents the state of the art in unsupervised keyphrase extraction. KP-Miner operates on n-grams and uses a modified version of tf-idf where the document frequency for n-grams with n greater than one is assumed to be one. We will explain the intuition behind this modification later as we have adopted it in our algorithm as well. KP-Miner's initial candidates are comprised of the longest n-grams that do not contain a stop word or punctuation mark, occur for the first time within the first 400 words of the document and have a term frequency above a minimum threshold determined by document length. KP-Miner also boosts the weights of multi-word candidates in proportion to the ratio of the frequencies of single word candidates to all candidates. In a reranking step, the tf-idf of each term is recalculated based on the number of times it is subsumed by other candidates in the top 15 candidates list. Another tf-idf based unsupervised system is KX-FBK (Pianta & Tonelli, 2010) which uses some of the same heuristics as KP-Miner but with different formulations and was shown to underperform in comparison in the Semeval 2010 keyphrase extraction task.

An approach fundamentally different from tf-idf and its family of algorithms is TextRank. It is based on the intuition that 1) keywords in a document are more semantically interrelated as they are generally about related topics and 2) that semantic relatedness can be estimated using co-occurrence relations. Therefore in TextRank a graphical representation of the text is constructed in which edges connect words co-occurring in a window of a certain length. The PageRank algorithm is then applied to this network of words to distinguish the important ones which are then reassembled into phrases wherever they occur next to each other in the text.

TopicRank (Bougouin et al., 2013) which to the best of our knowledge is the state of the art in graph-based keyphrase extraction, is an enhancement of TextRank. Here, nodes represent topics which consist of sets of candidate terms clustered around shared sub-terms. In (Liang et al., 2009) Chinese search engine query logs are used to extract candidate terms which are used as nodes in the graph. Edges are weighted based on co-occurrence count. Also candidate terms which are longer or whose first occurrence is in the title or first paragraph have boosted edge weights. SingleRank (Wan & Xiao, 2008) also uses co-occurrence counts as edge weights. It ranks noun phrases in the text based on the sum of their word weights. ExpandRank (Wan & Xiao, 2008) builds upon SingleRank by incorporating neighboring documents but without significant performance improvements (Hasan & NG, 2010).

3 Method

Our algorithm processes an input document in four stages. In the first stage we extract all possible n-grams from the input text and eliminate those that are highly unlikely to be keyphrases, for instance n-grams containing punctuation marks. In the second stage the remaining n-grams are ranked based on a modified version of tf-idf. In the third stage the top ranking candidates from stage two are reranked based on additional statistical heuristics such as position of first occurrence and term length. In the fourth and final stage the ranking produced in stage three is incorporated into a graph-based algorithm which produces the final ranking of keyphrase candidates.

3.1 Eliminating Unlikely Candidates

In the first stage all possible n-grams in the text for n from 1 to 6 are produced. Those n-grams considered highly unlikely to be keyphrases are eliminated from the candidates list. These include n-grams containing stop words, punctuation marks or words whose part of speech tag is anything different than noun, adjective or verb. Furthermore n-grams whose frequency of occurrence in the text falls below a minimum threshold are also eliminated. In the current work this threshold is determined based on document length and is 0 for short documents, 2 for medium-length documents and 3 for long documents where short is defined as containing less than 1500 words, medium as between 1500 to 4000 and long as any document with more than 4000 words.

3.2 Initial Ranking of All Candidates

In the second stage n-grams not eliminated in the first stage are ranked based on a modified version of tf-idf as used in KP-Miner. The modification involves changing the document frequency count in idf calculation such that for n-grams with n > 1 document frequency is always considered to be one. In other words we assume that all multi-word candidates occur in one document only. This is because while rareness is a reliable indication of semantic importance in the case of single words, it does not offer the same accuracy when it comes to multi-words. In many cases relatively common single words can combine into rare multi-words without much semantic importance. For example, in the Semeval test dataset of 100 full-length academic papers, to be described later in the evaluation section, the n-grams *control has*, *rule satisfies* and *become known* all have a document frequency of 1. On the other hand phrases chosen by humans as keyphrases such as *Query expansion* which occurs in 9 documents and as a keyphrase in 4 or *language models* which occurs in 12 document and again in 4 of them as key, have relatively high document frequency counts. These examples demonstrate how including the actual document frequency counts in idf calculation could be disadvantageous for distinguishing multi-word keyphrases.

3.3 Reranking Top Candidates

At the end of stage two we have an initial ranking of our candidates based on their tf-idf scores. In the third stage we rerank the top T candidates from stage two based on additional heuristics. These heuristics are position of first occurrence, term length and subsumption count. In the current work T is set to 100 based on experiments on a small development set of 40 documents from the Semeval trial set.

The position of first occurrence heuristic has performed consistently well in previous keyphrase extraction experiments. Medelyan and Witten (2008) use a linear decay function of the position of first occurrence as a feature in their supervised algorithm. It has also been utilized in unsupervised methods. In KP-Miner a constant position threshold is used where n-grams whose first occurrence is beyond it are eliminated from the candidate list. KX-FBK uses the linear decay function raised to the power of two. We introduce a novel encoding of this heuristic in the form of a logarithmic decay function, which as we will show in the discussion section outperforms all aforementioned variations. We define the Position of First Occurrence factor (PFO) according to the following formula:

$$PFO(t,d) = \log\left(\frac{cutoffPosition}{p(t,d)}\right) \quad (1)$$

where *p(t,d)* is the position of term t's first occurrence in document d. In the current work *cutoffPosition* is set to 3000 as it performed best in experiments on the development set.

Regarding term length, we hypothesize that among words with a high likelihood of being a

keyphrase, in this case the top 100 candidates from stage two, the addition of a word to an n-gram is likely to construct a more semantically specific phrase geared towards signifying a specific topic or subject e.g. *web* versus *semantic web*. Therefore longer n-grams are generally more likely to be keyphrases. Accordingly, we boost term t's weight by its term length, TL(t) where length is the number of space separated words in t.

Finally, we recalculate a term's tf-idf weight by reducing its term frequency by its subsumption count among the top 100 candidates. Term t is said to be subsumed by term t_s when t_s contains t.

The following formula shows how the statistical weight for term t in document d is calculated:

$$w_s(t,d) = \big(tf(t,d) - subSumCount(t,d)\big) * idf(t) * PFO(t,d) * TL(t) \quad (2)$$

Where *subSumCount(t,d)* is the sum of term frequencies of all terms included in the top T list that subsume t.

3.4 Graph-based Ranking

In the fourth and final stage of our algorithm we use terms with positive weights after stage three as nodes in a graphical representation of the text. An edge is placed between two nodes if they co-occur within a window of width d. Whereas d is usually small, generally less than 20 words in most graph-based algorithms, we have chosen a large window of 1500 and instead attenuate the edge weight based on the average log decayed distance between all co-occurrences of the term pair as show in equation 3 below

$$w_d(t_i,t_j) = \frac{\sum_{i=1}^{tf(t_i)}\sum_{j=1}^{tf(t_j)} \log\left(\frac{winSize}{|pos_i - pos_j|}\right)}{numCo\text{-}occurrences(t_i,t_j)} \quad (3)$$

where *winSize* is the co-occurrence window size set to 1500 in the current work based on F-measure performance on the development set, pos_i and pos_j are the respective positions of occurrences of terms t_i and t_j and *numCo-occurrences*(t_i,t_j) is the number of co-occurrences of the terms within the window of 1500.

Furthermore, we incorporate term weights calculated using statistical features in the previous stages into the graphical representation of the text. We hypothesize that term weights calculated using statistical features may serve as a first estimate of a term's keyphraseness, i.e. likelihood of being a keyphrase. The PageRank algorithm simulates a random walker on the graph. Each node's eventual PageRank score reflects the portion of time the walker spends on that node (Langville & Meyer, 2011). To make sure terms with higher statistical weights are visited more often we would want higher transition probabilities between them but lower transition probabilities between terms with lower weights. Therefore we use the product of the term pair's weights as a factor into the weight of the edge between them in the graph. The weight of the edge between terms t_i and t_j is calculated using the following equation:

$$w_e(t_i,t_j) = w_d(t_i,t_j) * w_s(t_i) * w_s(t_j) \quad (4)$$

where, as previously defined in equation 3, w_d is the distance based portion of the edge weight while $w_s(t_i)*w_s(t_j)$ takes the terms' statistical properties into account.

For each node we normalize edge weights by dividing each outgoing edge weight by sum of outgoing edge weights for that node. This results in a slightly modified formula for PageRank compared to the one used in TextRank, where edges are uniform weight, as shown below.

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{w_e(j,i) * S(V_i)}{\sum_{k \in Out(V_j)} w_e(j,k)} \quad (5)$$

where $S(V_i)$ is the PageRank score of node V_i, d is the damping factor usually set to 0.15 and $In(V_i)$ and $Out(V_i)$ are the sets of edges where node V_i is the destination or the source respectively. PageRank consists of iteratively calculating the scores for each node until convergence where scores do not change significantly between iterations. The converged-on PageRank score for each node is our algorithm's final output and determines the rankings of the candidate terms.

4 Evaluation

We evaluate our keyphrase extraction algorithm by comparing it to two state-of-the-art algorithms, KP-Miner and TextRank, on three datasets: The Semeval 2010 keyphrase extraction shared task dataset, the Inspec dataset of ACM abstracts and the Krapivin dataset of full length papers. To obtain results for KP-Miner we have used an executable kindly shared with us by the system's author. For TextRank we have built on an existing open source implementation. The comparisons between the algorithms are done using the precision and recall at k metric where the top k

terms returned by each algorithm are used to measure precision and recall. Here k ranges from 1 to 15. We also calculate the F-measure for k = 5, 10 and 15. In the following section we describe each dataset in detail and report the results achieved by each algorithm on each dataset.

The Semeval and Inspec datasets have also been used by Bougouin et al. (2013) for evaluating their implementation of TextRank along with more advanced graph-based algorithms SingleRank and TopicRank. We have used these results for further comparisons between our method and advanced graph-based algorithms as reported in section 4.4.

4.1 Semeval Dataset

The Semeval dataset was used in the Semeval 2010 keyphrase extraction shared task (Kim et al., 2010). To the best of our knowledge this shared task is the largest recent comparison of keyphrase extraction algorithms and an algorithm's performance on this dataset is a relatively good indication of where it stands compared to others in the field. The Semeval dataset consists of 284 full length ACM articles divided into a test set of size 100, training set of size 144 and trial set of size 40 which we used as the development set for parameter tuning. Each article has two sets of human assigned keyphrases: the author-assigned and reader-assigned ones. The gold standard used in our experiments is the combined set of author and reader assigned keyphrases which is the same as was done in the Semeval shared task. The table below provides a statistical overview of this dataset's documents.

100 docs	Document Length	Number of Keyphrases	Keyphrase Length
Max.	14171	29	8
Avg.	7979	15.13	2.14
Min.	4060	9	1

Table 1. Semeval test set statistics.

We have compared our algorithm with KP-Miner and TextRank using only the 100 documents in the test set. The following diagram shows the average precision and recall achieved by each algorithm. As was done in the Semeval task, comparisons are done between once stemmed human assigned keyphrases and ranked candidates returned by each algorithm.

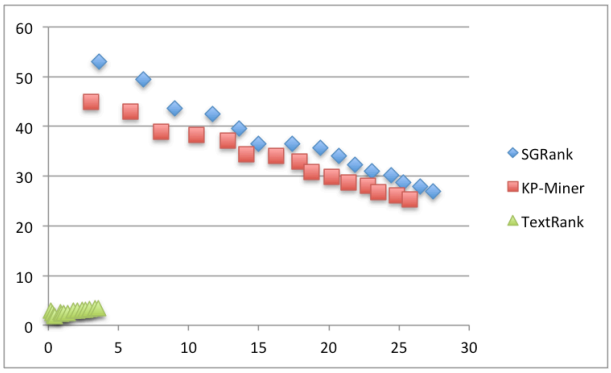

Figure 1. Semeval precision(y), recall(x) k < 16

The following table shows the achieved F-measure for each algorithm at k=5, 10 and 15. It also contains the corresponding percentage improvement at each k. The statistical significance of each improvement is measured using a 2-sided paired t-test. Improvements are in **bold** font where they are statistically significant at $p < 0.05$.

K =	5	10	15
SGRank	20.25	26.07	27.20
KPMiner Improvement	19.01 6.5%	24.06 **8.3%**	25.54 **6.4%**
Textrank Improvement	1.25 **1509%**	2.46 **960%**	3.47 **683%**

Table 2. Semeval F-measures and improvements.

As can be seen from the above results our method outperforms KP-Miner in both precision and recall for all k and achieves statistically significant improvements in the F-measure over KP-Miner for k=10 and 15. These results are noteworthy considering that in the Semeval keyphrase extraction shared task KP-Miner was the best performing unsupervised algorithm, and the second best overall out of 19 systems, outperforming prominent supervised algorithms such as Maui (Medelyan et al., 2009). TextRank seems to generally underperform on longer documents and has performed poorly on the Semeval dataset.

4.2 Inspec Dataset

The Inspec dataset is comprised of 2000 ACM abstracts divided into test, training and validation sets containing 500, 1000 and 500 abstracts respectively. We follow the same approach as

taken by Mihalcea and Tarau (2004). We use only the 500 documents in the test set. The following table provides a statistical overview of this document set.

500 docs	Document Length	Number of Keyphrases	Keyphrase Length
Max.	338	31	9
Avg.	121.8	9.8	2.3
Min.	23	2	1

Table 3. Inspec dataset statistics.

Figure 2 shows the average precision and recall for all three algorithms for k from 1 to 15. Table 4 shows the F-measure improvements made by our method over the two other algorithms for k=5, 10 and 15. As these results show, on this dataset of relatively short documents, TextRank outperforms KP-Miner for k>2. Our algorithm achieves higher precision and recall than both KP-Miner and TextRank for all k with statistically significant gains in the F-measure for k=5, 10 and 15.

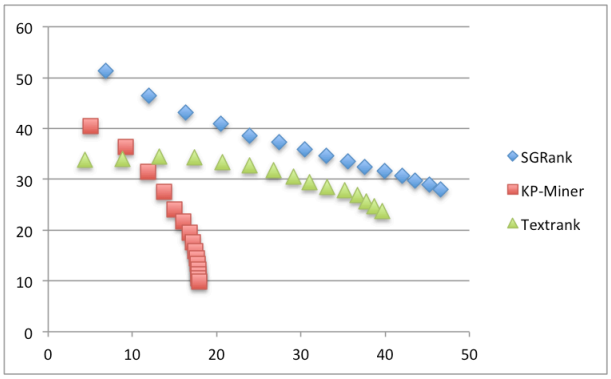

Figure 2. Inspec precision(y), recall(x) k < 16

K =	5	10	15
SGRank	29.16	33.95	33.66
KPMiner	18.45	15.89	12.73
Improvement	**59.7%**	**118%**	**175%**
TextRank	25.53	30.6	29.7
Improvement	**15.4%**	**13.3%**	**17.7%**

Table 4. Inspec F-measures and improvements.

4.3 Krapivin Dataset

The Krapivin dataset consists of 2000 full length ACM papers. This dataset has been prepared by Krapivin et al. (2009). Each article has author-assigned and editor-corrected keyphrases that we use as the gold standard in our evaluation. Our experiments are done on a 400-document subset of this dataset. The table below provides a statistical characterization of these 400 documents.

400 docs	Document Length	Number of Keyphrases	Keyphrase Length
Max.	16721	24	6
Avg.	7934	6.38	2.1
Min.	3892	1	1

Table 5. Krapivin dataset statistics.

On this dataset keyphrases and candidate terms have been stemmed once before comparison. Similar to the previously mentioned experiments we have measured the precision and recall of all three algorithms for k from 1 to 15 as shown in figure 3. Table 6 contains the F-measures for all three algorithms at k=5, 10 and 15 along with the improvements made by our algorithm. Similar to the Semeval dataset TextRank performs very poorly on this dataset of longer documents. KP-Miner performs much better but both methods are outperformed by our method on all k with statistical significance as shown in Table 6.

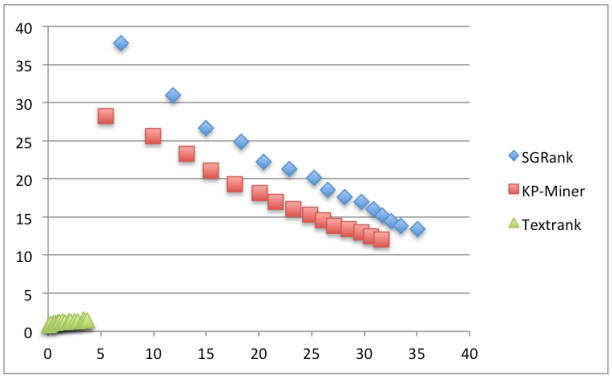

Figure 3. Krapivin precision(y), recall(x) k < 16

K =	5	10	15
SGRank	21.2	21.6	19.4
KPMiner	18.43	18.65	17.4
Improvement	**15.%**	**16.1%**	**11.7%**
Textrank	1.02	1.61	2.1
Improvement	**1974%**	**1240%**	**823%**

Table 6. Krapivin F-measures and improvements.

4.4 Advanced Graph-based Methods

As mentioned previously Bougouin et al. (2013) introduce TopicRank and use F-measure at k=10 to compare against TextRank and another advanced graph-ranking method SingleRank. They use the Semeval and Inspec datasets for comparison providing us with an opportunity to compare our performance with those of TextRank and the two more advanced graph-based algorithms. Table 7 contains the F-measures at k=10 for our algorithm, SGRank, and all aforementioned algorithms. Note that the particular implementation of TextRank used in this paper performs worse than ours on the Inspec dataset but better for the Semeval dataset.

F at k=10	Inspec	Semeval	Average
SGRank	33.95	26.4	30.1
TextRank	12.7	5.6	9.1
SingleRank	35.2	3.7	19.4
TopicRank	27.9	12.1	20

Table 7. Comparison with Advanced Graph-based methods F-measures at K=10.

As seen in Table 7 our algorithm's average performance is considerably better than all of the advanced graph-based algorithms.

5 Discussion

As shown in the preceding results our algorithm outperforms all other methods in all the used datasets. The only exception is SingleRank which marginally outperforms our method on the Inspec dataset but performs very poorly on the Semeval dataset, as seen in Table 7. Also worth noting is KP-Miner's relatively poor performance on the shorter documents of the Inspec dataset. This could potentially be due to the fact that KP-Miner only considers terms as candidates which occur on their own in the text i.e. surrounded by punctuation marks or stop words. In shorter documents it is more likely that fewer keyphrases would occur in such conditions in the text, causing them to be eliminated early on by KP-Miner. Our algorithm however considers all n-grams without requiring that they occur on their own. This allows us to consider more candidates and avoid a performance reduction in shorter documents. However, there is an advantage to eliminating terms that never occur on their own. Many keyphrases are multi-words. In some cases smaller parts of keyphrases tend to occur in high frequencies, as they are related to the topic of the document and are sometimes used in place of the keyphrase, and therefore achieve high rankings. We call such frequent sub-phrases keyphrase fragments. For example document C-1 in the Semeval test set includes two keyphrases *grid service discovery* and *web service* leading to a highly ranked keyphrase fragment *service*. High ranking keyphrase fragments are detrimental to the algorithm's performance. One way to counteract them is based on the observation that they rarely occur on their own as they usually appear as part of larger phrases. This is the motivation behind KP-Miner's elimination of candidates that do not occur on their own. Therefore, to consider all candidates, while countering the keyphrase fragments problem, we calculate the subsumption count over a much larger portion of the ranked terms compared to KP-Miner. This larger list will include more terms which keyphrase fragments are a part of, causing greater reductions in the fragments' rankings. The number of top candidates used in KP-Miner to calculated subsumption is set equal to an input parameter that determines the number of keyphrases to be returned to the user. In the current work and the Semeval shared task this parameter is 15. In other words KP-Miner calculates the subsumption count over the top 15 terms whereas we calculate it over the top 100 terms. To test the effectiveness of this strategy we reduced our subsumption threshold to 15. This change led to a 9% decrease in the F-Measure at k=15 on the Semeval dataset, 4.7% decrease on the Inspec and 1.5% decrease in the Krapivin dataset. Note that for the rest of the Discussion section percentage changes are those of the F-measure at k=15. Our algorithm's high performance on both short and long documents indicates the viability of considering all n-grams as candidates and mitigating the effect of keyphrase fragments by counting subsumption over more top ranking terms.

Another novel aspect of our algorithm is its formulation of the position of first occurrence heuristic as described by the PFO function in equation 1. We compare our approach with two other unsupervised algorithms that utilize this heuristic: KP-Miner and KX-FBK. The method used in KP-Miner is a hard cutoff threshold where candidates whose first occurrence is beyond 400

words into the document are eliminated. KX-FBK uses the following decay function:

$$PFO_{kx}(t,d) = \left(\frac{document\ length - p(t,d)}{document\ length}\right)^2 \quad (6)$$

We tested our system with our PFO function replaced by those of KP-Miner and KX-FBK on the Semeval, Inspec and Krapivin datasets. Replacement with KX-FBK's PFO led to respective reductions in the F-Measure at k=15 of 6.5%, 9.5% and 15.1%. Replacement with KP-Miner's PFO led to a 10.9% reduction in Semeval but no reduction in Inspeq and a 2.9% improvement in Krapivin. We also replaced our PFO function with the linear decay function used by Medelyan and Witten (2008). This function is the same as equation 9 but without the exponent. This led to a 1.4% reduction in Semeval, 0.7% reduction in Inspec and a 2.9% reduction in Krapivin. These results show that our encoding of the PFO heuristic as a logarithmic decay function leads to overall gains in accuracy although it underperforms slightly compared to KP-Miner's PFO on the Krapivin dataset which points to further room for improvement. One possible future direction would be to design functions that adjust the *cutoffPosition* in equation 1 based on document length as some sensitivity to this was observed in our experimentations. We also replaced our term length factor, TL in equation 2, with KP-miner's boosting function for multi-words. This caused a 3.2% reduction in Semeval, 5.1% reduction in Inspec and 2.3% reduction in Krapivin.

Our algorithm uses graph-based methods on top of statistical features to capture keyphrases not distinguishable using statistical heuristics. To test the effectiveness of this addition we eliminated the graph-based reranking stage. This caused a 1.1% reduction in Semeval, 4% reduction in Inspec and a 4.3% reduction in Krapivin which demonstrates that our approach of combining statistical and graph-based features leads to overall improvements in performance. Our method also introduces a novel distance-based edge weighting formula to the graph-based family of algorithms. Most graph-based algorithms place edges where terms co-occur within a window of a few words. This is equivalent to a sudden drop in the estimation of semantic relatedness at the edge of the window. We however choose a much larger window of 1500 and gradually reduce the edge weight with increasing distance between the terms according to equation 3. To measure the effectiveness of this approach we compared it with a window of 100 words with no positional decay, i.e. w_d in equation 4 is set to one for terms occurring within the window of 100 and zero otherwise. This caused a 2.2% drop in the Semeval dataset i.e. it performed lower than with no graphical reranking at all. In Krapivin it caused a 2.4% drop in performance and a 0.4% drop in the Inspec dataset. For further comparison we replaced w_d with the *dist* function used in TopicRank which is the sum of inverse distances between all occurrences of a term pair. This caused a 1.4% reduction in Semeval, no difference in Krapivin and a 0.3% reduction in Inspec. These results demonstrate the effectiveness of our novel distance based edge weighting function.

An interesting point is that both our positional functions, PFO and w_d are logarithmic decays. This hints at a logarithmic decrease in semantic importance or relatedness with increased distance which is the same as how the idf function relates a word's semantic importance to its document frequency. Our initial hypothesis for the success of logarithmic decay functions is that both positional and document frequency heuristics are governed by the law of diminishing returns. That is, the distinguishing power of each heuristic decreases as the inputs increase. Taking the document frequency heuristic (df) as an example, we know that rareness, i.e. small dfs, indicate higher semantic importance. Therefore, in a corpus of 1000 documents, a word with a df of 1 is much more likely to be a keyphrase than a word with a df of 20, as reflected in their idf scores. However as the df increases the same difference in df's does not imply the same difference in probability of being a keyphrase e.g. we intuitively know that based on rareness alone, our estimate of the difference in the probability of being a keyphrase for a pair of terms with df's 980 and 1000 would be much less reliable compared to a pair with df's 1 and 20, even though the difference in the df pairs are the same. The same logic applies to the position of first occurrence and distance based semantic relatedness heuristics. Incorporating this diminishing returns property into the mathematical formulation of the heuristic calls for a function with a decreasing absolute value slope i.e. with a second derivative with the opposite sign of the

first, to reflect our decreasing confidence in the heuristic as values increase. This rules out linear functions. Although other functions such as reciprocals fulfill this property, judging based on the success of idf and our positional decay functions it seems that logarithmic decay does better at modeling the intrinsic rate of this diminishing distinguishing power of the heuristic, perhaps due to its slower decline. Why this is and whether better decay functions can be designed or tuned to specific domains is a future direction we plan to explore. It is also worth noting that unlike most graph-based algorithms whose performances are completely dependent on a POS tag filter, SGRank suffers relatively slight reductions in F-measure at 15 without the POS tag filter: 3.6% on Semeval, and 2.6% in Krapivin and 24.5% on Inspec.

Feature Tested	Datasets Average F at 15	Datasets Average Change	S.	I.	K.
None	26.7	0	27.2	33.6	19.4
Subsume Top 15	25.3	-5.6%	24.7	32	16.4
kx-fbk PFO	24.1	-10.3%	25.4	30.4	16.4
kp-miner PFO	25.95	-2.6%	24.2	33.6	19.9
kp-miner multi-word boosting	25.74	-3.5%	26.3	31.9	18.9
edge weight =1 if d<100	26.3	-1.6%	26.6	33.5	18.9
topicRank edge weighting	26.5	-0.56%	26.8	31.9	18.9
no POS-tagging	23.5	-10.2%	26.2	25.4	18.9
no graph-based	25.9	-3.1%	26.9	32.2	18.5

Table 8. Effects of individual features on performance. Columns S., I. and K. contain F-measures (k=15) for the Semeval, Inspec and Krapivin datasets respectively.

Table 8 contains a summary of how the elimination or replacement of different features affects the performance of our algorithm, as discussed previously. It contains the F-measure at k = 15, averaged across all datasets, for the full algorithm along with variations of it produced by changing different features.

6 Conclusion and Future Directions

We introduce an unsupervised keyphrase extraction algorithm that combines statistical and graph-based heuristics and is able to improve upon the state of the art, with statistical significance, on several datasets. Among other features, our algorithm uses a novel variation of the subsumption heuristic. We also demonstrate the suitability of log decay functions for mathematically expressing heuristics that are based on phrase distance such as the position of first occurrence and the weighting of graph edges based on the average distance of phrase occurrences. Another way of looking at the presented algorithm is as a term weighting scheme. Therefore an interesting future direction would be to investigate whether replacing traditional term weighting schemes, e.g. tf-idf, in areas such as information retrieval, document clustering and supervised algorithms where tf-idf is used as a feature would cause any improvements in performance.

7 References

Bougouin A., F. Boudin, and B. Daille. 2013, October. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International Joint Conference on Natural Language Processing (IJCNLP),* pages 543-551.

D'Avanzo F., B. Magnini, and A.Vallin. 2004. Keyphrase extraction for summarization purposes: The LAKE system at DUC-2004. In *Proceedings of the 2004 document understanding conference*.

El-Beltagy S. and A. Rafea. 2009. KP-Miner: A keyphrase extraction system for English and Arabic documents. In *Information Systems*, *34*(1), pages 132-144.

Samhaa R. El-Beltagy and Ahmed Rafea. 2010. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 190-193.

Fortuna B., M. Grobelnik, and D. Mladenić. 2006. Semi-automatic data-driven ontology construction system. In *Proceedings of the 9th international multi-conference information society,* pages 223–226.

Hammouda K., D. Matute, and M. Kamel. 2005. Corephrase: Keyphrase extraction for document clustering. In *Machine Learning and Data Mining in Pattern Recognition*, pages 265-274.

Hasan, K. S., & Ng, V. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (pp. 365-373). Association for Computational Linguistics.

Hulth A. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216-223.

Jones K.. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, *28*(1), Pages 11-21.

Kim S., O. Medelyan, M. Kan, and T. Baldwin. 2010. Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles. In *Proceedings of the 5th International Workshop on Semantic Evaluation* , pages 21-26.

Krapivin M., A. Autaeu & M. Marchese. 2009. Large dataset for keyphrases extraction. Technical Report DISI-09-055.

Langville, A. N., & Meyer, C. D. 2011. *Google's PageRank and beyond: the science of search engine rankings*. Princeton University Press.

Leake D., A. Maguitman, and T. Reichherzer. 2003. Topic Extraction and Extension to Support Concept Mapping. In *FLAIRS Conference*, pages 325-329.

Liang, W., Huang, C., Li, M., & Lu, B. L. 2009. Extracting Keyphrases from Chinese News Articles Using TextRank and Query Log Knowledge. In *PACLIC*

Lops P., M. Gemmis, and G. Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73-105.

Medelyan, O., & Witten, I. H. 2008. Domain Independent Automatic Keyphrase Indexing with Small Training Sets. *Journal of the American Society for Information Science and Technology*, *59*(7), 1026-1040.

Medelyan, O., Frank, E., & Witten, I. H. (2009). Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3* (pp. 1318-1327). Association for Computational Linguistics.

Mihalcea R. and P. Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of International Conference on Empirical Methods in Natural Language Processing*, pages 404–411

Page, L., Brin, S., Motwani, R., & Winograd, T. 1999. The PageRank citation ranking: Bringing order to the web.

Pianta, E., & Tonelli, S. 2010. KX: A flexible system for keyphrase extraction. In *Proceedings of the 5th international workshop on semantic evaluation* (pp. 170-173). Association for Computational Linguistics.

Qiu M., Y. Li, and Jing Jiang. 2012. Query-oriented keyphrase extraction. In *Information Retrieval Technology*, pages 64-75.

Salton, G., A. Wong, and C. Yang. 1975. A vector space model for automatic indexing. In *Communications of the ACM*, *18*(11), pages 613-620.

Xiaojun Wan and Jianguo Xiao. 2008. Single Document Keyphrase Extraction Using Neighborhood Knowledge. *In Proceedings of the 23rd National Conference on Artificial Intelligence* - Volume 2 , pages 855–860. AAAI Press.