Corpus-independent Generic Keyphrase Extraction Using Word Embedding Vectors

Rui Wang
The University of Western Australia
35 Stirling Highway, Crawley, W.A. 6009, Australia
rui.wang@research.uwa.edu.au

Wei Liu*
The University of Western Australia
35 Stirling Highway, Crawley, W.A. 6009, Australia
wei.liu@uwa.edu.au

Chris McDonald
The University of Western Australia
35 Stirling Highway, Crawley, W.A. 6009, Australia
chris.mcdonald@uwa.edu.au

ABSTRACT

Keyphrase extraction from a given document is a difficult task that requires not only local statistical information but also extensive background knowledge. In this paper, we propose a graph-based ranking approach that uses information supplied by word embedding vectors as the background knowledge. We first introduce a weighting scheme that computes informativeness and phraseness scores of words using the information supplied by both word embedding vectors and local statistics. Keyphrase extraction is performed by constructing a weighted undirected graph for a document, where nodes represent words and edges are co-occurrence relations of two words within a defined window size. The weights of edges are computed by the afore-mentioned weighting scheme, and a weighted PageRank algorithm is used to compute final scores of words. Keyphrases are formed in post-processing stage using heuristics. Our work is evaluated on various publicly available datasets with documents of varying length. We show that evaluation results are comparable to the state-of-the-art algorithms, which are often typically tuned to a specific corpus to achieve the claimed results.

Keywords

keyphrase extraction, word embedding vectors, unsupervised algorithm

1. INTRODUCTION

Keyphrases provide a high level description of a document, which are is important to many areas of text processing, such as document classification and clustering, information retrieval, and question-answering. Automatically extracting keyphrases from large document sets is a desirable but difficult task. The state-of-the-art algorithms for keyphrase extraction currently are only able to achieve a precision of no more than 35% even on their chosen datasets [16]. In contrast to other Natural Language Processing (NLP) tasks, keyphrase extraction techniques desire much more room for improvement.

Keyphrase extraction is difficult because it requires not only the information contained in a document or a document set, namely *local statistical information*, but also extensive *background knowledge* about the world. Early approaches [20, 27, 29] employ only *local statistical information*. The shortcoming of the *local statistical information* is that it does not contain semantic information about the words appearing in a document. Thus, words are treated as solely statistical elements in these approaches.

Further improvements have been made by recently proposed knowledge based approaches that employ semantic relation information from external knowledge bases, such as Wordnet and Wikipedia, to provide the *background knowledge*. Liu et al. [23] use a clustering-based approach that clusters semantically related candidate phrases using statistical information from Wikipedia. Grineva et al. [12] also use Wikipedia for weighting terms and determining semantic relatedness between them. Wang et al. [35] use the PageRank algorithm with WordNet as a semantic network for background knowledge.

Although the approaches using semantic knowledge bases have shown their efficiencies and improvements, identifying keyphrases requires more background knowledge than just semantic relation information. Intuitively, a phrase is said to be a keyphrase because, firstly, it captures important information about a document and, secondly, it is a semantically and syntactically correct phrase without unnecessary words. Thus, two measures are considered when selecting a keyphrase: how much information a phrase can contribute towards the main ideas of a document, and the likelihood that a sequence of words can be considered as a phrase. Tomokiyo and Hurst [31] refer to these two measures as *informativeness* and *phraseness*, which are similar to the terms *termhood* and *unithood* employed in the text mining field [36, 37].

In this paper, we propose a graph-based ranking model using the information supplied by distributed word representa-

*Corresponding author

tions as background knowledge for identifying informativeness, and local mutual information for identifying phraseness. The distributed word representations are also known as *word embeddings* [32], where a word is represented as a low dimensional real-valued vector. The values of each vector are trained using unsupervised machine learning algorithms, enabling the use of large-scale training sets, such as Wikipedia snapshots [6].

Our model represents a document as a weighted undirected graph, where vertices represent words, and edges are the co-occurrence relations between words constrained by a window size or sentence boundary. Informativeness and phraseness scores of words are defined and computed using the values provided by the word embeddings and local statistical information. The computed scores are then assigned to each edge as weights. We adopt a weighted PageRank algorithm to rank the words and finally apply heuristics to form phrases. Our model is evaluated using datasets of short, medium, and long documents. The result demonstrates that the performance is comparable to state-of-the-art algorithms. SENNA [6] word embeddings are used in our evaluation, that are pre-trained over Wikipedia.

This paper is organised as follows: Section 2 briefly summarises the related work, and Section 3 briefly describes the word embeddings. Section 4 introduces our word attraction weighting scheme, and Section 5 describes our graph-based ranking algorithm. Section 6 presents implementation details, and experiments are described in Section 7. We finally draw our conclusion with an outlook to future work in Section 8.

2. RELATED WORK

Generally, keyphrase extraction techniques can be classified into two groups: supervised machine learning approaches and unsupervised ranking approaches.

Supervised machine learning approaches treat the keyphrase extraction as a classification problem, in which a candidate must be classified as either a keyphrase or not. A classifier needs to be trained using annotated training data. The trained model is then applied to documents for which keyphrases are to be identified. The earliest two studies are presented by Frank et al. [11] and Turney [33], where the *Naïve Bayes* classifier and *C4.5* decision tree algorithms are used for classifying keywords. Hulth [19] used a combination of lexical and syntactic features where noun phrases and part-of-speech tag patterns were used to help identify candidates. Following Hulth's approach, more features have been added for training classifiers. In recent studies, Yih et al. [39] use *Term Frequency Inverse Document Frequency*, web page meta data, and query log files to train the classifier, and Ercan and Cieckli [10] add lexical chains in the training.

Unsupervised approaches can be grouped into statistical-based and graph-based approaches. Statistical-based approaches typically represent texts as matrices, then apply statistical techniques to rank the words. A well-known approach from this stream of studies is the Term Frequency Inverse Document Frequency (TF-IDF) [20]. It is based on the idea that a term occurring frequently and is distributed evenly in a corpus is not a good discriminator, and thus should be assigned less weight than a term that only occurs frequently in a few particular documents. Matsuo and Ishizuka [26] present a study that applies χ^2 squared test on word co-occurrence distribution without using a corpus. Graph-based approaches were introduced in the late 1990s [27, 29]. These approaches represent a document as a graph where vertices represent words, and edges are connected based on either lexical or semantic relations, such as a co-occurrence relation. Ohsawa et al. [29] presented a study using both graph clustering and statistic-based scoring algorithms to rank the keyphrases. In the last decade, many studies have employed the PageRank algorithm [5] and its variants for keyphrase ranking. Mihalcea and Tarau [27] first demonstrated the efficiency of using PageRank for keyphrase extraction. Following their approach, many studies proposed using PageRank or variants with different weighting schemes. Wang et al. [35] used the semantic relatedness identified from WordNet as the weights assigned to the graph. Wan and Xiao [34] incorporate the co-occurrence information from a set of similar documents into the graph weights. For a given document d, first they identify the k nearest neighbour documents from either the document sets or online documents as D, then build a graph using the co-occurrence information as the weights collected from both d and D.

3. WORD EMBEDDINGS

In this section, we briefly review word representation techniques and word embeddings. In many early rule-based and statistical NLP studies, words were represented as atomic symbols – just the words themselves, e.g. `dog`, `cat`. In early machine learning approaches for NLP, words were represented as unique identifiers referring to feature vectors using one-hot representations, in which each vector has the same length as the size of the vocabulary with only one dimension asserted to indicate the ID of the word. For example: the word `dog` may be represented as [0 0 0 0 0 1 0 0 0 0] in a vocabulary of 10 words. Obviously, one-hot representations suffer from the *curse of dimensionality* when one tries to model the joint distribution over a large vocabulary (dimensions). One-hot representations are also unable to represent any example not appearing in the training set, known as *data sparsity*. Neither one-hot representations nor atomic symbol representations are capable to capture semantic relations or similarities between words. Another approach is to represent words in a co-occurrence matrix of size $W \times C$, where W is the size of a vocabulary and C is the set of the co-occurrence contents, such as documents, sentences, or words. In this representation, one can analyse the distribution of words in order to capture the semantic relations and similarities between words, also known as *distributional semantic modelling*, based on the theory of distributional hypothesis [14]. Studies have shown that the co-occurrence matrix is able to capture semantic information while efficiently reducing the dimensionality of the matrix [4, 7, 25].

Distributed representation, introduced by Hinton [17, 18] in the 1980's, represents words or objects with references to cognitive representations. The idea is that human brains represent objects efficiently by characterising them using many features, and learn new objects by grouping them that are similar to known ones. The distributed representation of an object or a symbol is a vector of features that cap-

tures and characterises the meaning of the object or the symbol [1].

In NLP, distributed representations of words, called *word embeddings*, are low dimensional and real-valued vectors. Each dimension in the vector represents a feature of a word, and the vector can, in theory, capture both semantic and syntactic features of the word. Word embeddings are typically 50 to 200 dimensional vectors; for example the word `dog` may be represented as $[0.435, 0.543, ..., 0.128]$. Surprisingly, well-trained word embeddings encode far more semantic and syntactic information than people first realised. For example, Mikolov et al. [28] show that the result of embedding vector calculations show that `vec(king)` − `vec(man)` + `vec(woman)` ≈ `vec(queen)`, and `vec(apple)` − `vec(apples)` ≈ `vec(car)` − `vec(cars)`.

Bengio et al. [2] present a probabilistic neural language model that demonstrates how word embeddings can be induced with neural network probability predictions. The goal of the model is to compute the probability of a word given previous ones, and the embeddings may be seen as a 'side product' of the model. In an *n-gram*, the probability of the n word given preceding ones is $P(w_t|w_{t-n+1}, ...w_{t-1})$. Each word w_{t-i} is mapped to a word embedding vector $C(w_{t-i})$. By concatenating $n-1$ word embedding vectors, the probabilistic prediction of word n can be calculated using a probabilistic classification with a *softmax* activation function. After training, this model can potentially generalise the contexts not appearing in the training set. For instance, if one wants to calculate the probably of $p(running|the, cat, is)$, but the example of `the cat is running` does not appear in the training set, the model will generalise `cat` to `dog` if all of `the dog is running`, `the cat is eating`, and `the dog is eating` are in the training set.

However, the training speed for the above model is penalised by a large dictionary size. Collobert and Weston [6] present a discriminative and non-probabilistic model using a pairwise ranking approach. The idea is that the network assigns higher scores for correct phrases than the incorrect ones. For an *n-gram* from the dataset, the model first concatenates the embedding vectors as a positive example, then a corrupted *n-gram* is constructed as a negative example by replacing the middle word of the positive example with a random one from the vocabulary. The network is trained with a ranking-type cost:

$$\sum_{s \in S} \sum_{w \in D} max(0, 1 - f(s) + f(s^w))$$

where S is the set of windows of a text, D is the dictionary, f is the network, s is the positive example, and s^w is the negative one. Collobert and Weston train the embeddings over the entire English Wikipedia and show superior performance on many NLP tasks using the embeddings as initial values for their deep learning language model. In this paper, we use the word embeddings pre-trained by Collobert and Weston.

4. WORD ATTRACTION SCORE

In this section, we introduce a new scheme to compute the informativeness and phraseness scores of words using the information supplied by both word embedding vectors and local statistical information. We refer to this as *word attraction score*.

Word attraction measures both informativeness - the importance of a word towards the core ideas of a document, and phraseness - the likelihood that a sequence of words can be considered as a phrase. We compute the importance of words by ranking their semantic relationship strength, and the likelihood of words being a phrase by calculating their mutual word information.

We consider that the key ideas of a document can be captured by a group or groups of words having strong semantic relationships. No words appear in a document just by chance, and semantically related words tend to appear together more frequently than unrelated ones. In a document, there are always a few important words that tie and hold the entire text together [13]. Thus, we can assume that the words having stronger semantic relationships attract each other in a document, and constitute the frame of the document. Other supporting words, having weaker relationships, are also attracted by the core words to form the body of the document.

We use the values contained in word embedding vectors to measure the semantic relatedness between words. Word embeddings capture semantic relations can be well explained by Harris' distributional hypothesis [14]: words that have similar distributional patterns tend to resemble each other in meaning. Therefore, the longer the distance of two word embedding vectors, the weaker their semantic relationship.

However, solely semantic strength of two words does not provide enough information to measure the importance of the words. Intuitively, two words cannot be said to be important to a document if both of them have very low frequencies, even though they may have a very strong semantic relation. Conversely, two words may be said important if 1) one of them has distinct high frequency whereas another has very low frequency, and 2) they have a very strong semantic relation. For example, if we see words `fishing` and `saltwater` only once in an article talking about travel, then they are unlikely to be important towards the core theme of the article. But assume that we see the word `fishing` 100 times, then the word `saltwater` would be a good candidate, even though it may only appear twice in the article. We introduce an equation to specifically cope this intuition, inspired by the Newton's law of universal gravitation. The idea is to assign higher scores to two potentially important words. Newton's law of universal gravitation states that any two objects in the universe attract each other with a force that is directly proportional to the product of their masses and inversely proportional to the square of the distance between them. We therefore consider that the word also attract each other with a force, where the masses of two objects in the law are interpreted as the frequencies of two words, and the distance between two words is calculated as the Euclidean distance of the word embedding vectors.

Formally, the attraction force of word w_i and w_j in a document D is computed as:

$$f(w_i, w_j) = \frac{freq(w_i) \times freq(w_j)}{d^2} \quad (1)$$

where $freq(w)$ is the occurrence frequency of the word w in D, d is the Euclidean distance of the word embedding vectors for w_i and w_j.

The likelihood that a sequence of words can be considered as a phrase, is calculated by their local mutual frequencies. Words tend to co-occur in common patterns that form phrases to represent concrete meanings. The higher the occurrence frequency a pattern has, the more likely that that pattern can be considered as a phrase. The dice coefficient [8] traditionally is used to compare the similarity of two samples. It is brought to the NLP field to measure the probability of two words co-occurring in a pattern, or by a chance [30]. Formally, a document D is a sequence of words w, represented as $D = \{w_0, w_1, ...w_n\}$. The dice coefficient is calculated as:

$$dice(w_i, w_j) = \frac{2 \times freq(w_i, w_j)}{freq(w_i) + freq(w_j)} \quad (2)$$

where $freq(w_i, w_j)$ is the co-occurrence frequency of words w_i and w_j, and $freq(w_i)$ and $freq(w_j)$ are the occurrence frequencies of w_i and w_j in D.

The attraction score is computed as the product of *dice* score which measures the *phraseness*, and *attraction force* which measures the *informativeness*, as

$$attr(w_i, w_j) = dice(w_i, w_j) \times f(w_i, w_j) \quad (3)$$

5. GRAPH-BASED RANKING

In this section, we introduce our ranking algorithm. Similar to many graph-based ranking algorithms, we represent a document as a weighted undirected graph, where vertices correspond to words, and edges represent co-occurrence relations between two words. Two vertices are connected if they co-occur within a distance constrained by a window size or a sentence.

5.1 PageRank

The original PageRank algorithm [5] ranks vertices of a directed unweighted graph. In a directed graph $G = (V, E)$, let $in(v_i)$ be the set of vertices that point to a vertex v_i, and $out(v_i)$ be the set of vertices to which v_i point. The score of v_i is calculated by PageRank as

$$S(v_i) = (1 - d) + d \times \sum_{j \in in(v_i)} \frac{1}{|out(v_j)|} S(v_j)$$

where d is the *dumping factor*, typically set to 0.85.

5.2 Weighted PageRank

In the original PageRank algorithm, the edges are unweighted. Xing and Ghorhani [38] introduce a weighted PageRank algorithm, where the graph remains directed with weights calculated by the number of linkages a vertex receives from, or sends to, other vertices. In the same year, Mihalcea and Tarau [27] introduce a simplified version for ranking texts, where the graph is undirected. In an undirected graph, the in-degree of a vertex equals to the out-degree of the vertex. The score of a vertex v_i can be calculated as:

$$WS(v_i) = (1 - d) + d \times \sum_{v_j \in in(v_i)} \frac{w_{ji}}{\sum_{v_k \in out(v_j)} w_{jk}} WS(v_j)$$

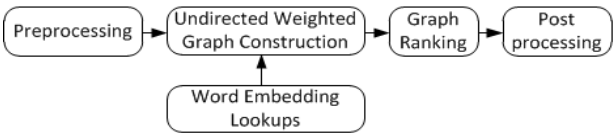

Figure 1: System Processing Pipeline

Figure 2: Preprocessing Pipeline

where w_{ij} is the strength of the connection between two vertices v_i and v_j.

In this paper, we use an equation derived from Mihalcea and Tarau's [27] ranking algorithm. In weighted PageRank, more weight assigned to a edge, higher scores the two vertices will have. Formally, in graph $G = (V, E)$, where V is the collection of all vertices, and E is the collection of all edges, let $C(v_i)$ be the set of vertices that edge to v_i. The score of v_i is calculated

$$S(v_i) = (1 - d) + d \times \sum_{v_j \in C(v_i)} \frac{attr(v_i, v_j)}{\sum_{v_k \in C(v_j)} attr(v_j, v_k)} S(v_j) \quad (4)$$

where $attr(v_i, v_j)$ is the attraction score between two vertices V_i and V_j calculated using Equation 3.

6. SYSTEM IMPLEMENTATION

In this section, we describe our implementation in detail. Our model uses a pipeline architecture, as presented in Figure 1. Firstly, we preprocess input texts using tokenising, part-of-speech tagging, and plural removal. Secondly, a weighted undirected graph is constructed. Vertices are input words from preprocessing, edges represent the co-occurrence relations between words, and the weight of an edge is computed by our weighting scheme. We then rank the graph using the weighted PageRank algorithm. In post processing, words are collapsed as phrases if they are adjacent in the text, and the phrases with highest scores are extracted as keyphrases.

The system is implemented using *Python 2.7* and the *Natural Language Toolkit* [3]. For part-of-speech (POS) tagging, we use the Stanford parser [21].

6.1 Preprocessing

The preprocessing pipeline is presented in Figure 2. We first split a text into sentences and tokenise them, and then we run POS tagging. After that, we remove plurals for tagged nouns. Finally, only adjectives and nouns are considered as valid words for ranking because the majority of keyphrases only contain adjectives and nouns. A threshold can also be used to filter out low frequency words.

Stemming is not performed at preprocessing stage, firstly because words with the same stem may represent different meanings and, secondly, the word embeddings do not contain word stems.

6.2 Word Embedding

Training word embeddings is computationally expensive. In this study, we use pre-trained embeddings[1] by Collobert and Weston [6]. They trained their model to discriminate a classification task: a word in the middle of an input is either related to the context or not. A positive example is an input from the corpus, and a corrupted one is generated by replacing the middle word of the positive example by a random word from the vocabulary as a negative example. The training used a window size of 11, and each embedding vector has a size of 50 dimensions. The embedding vectors were trained over 852 million words and the actual embeddings contain 130,000 common word vectors.

6.3 Graph Construction and Ranking

After preprocessing, all valid words are represented in the graph as vertices. In our model, we consider two words to have co-occurred if they appear in the same sentence. Two vertices are connected if a co-occurrence relation is found, and a weight is computed (i.e. attraction score), using Equation 3, and assigned to the edge. Once the graph is constructed, we use Equation 4 to score each vertex.

6.4 Post processing

After ranking, each word in the graph is assigned a score computed by the algorithm, and all the words and scores are saved into a dictionary D. We then adopt the same phrase formation approach proposed by [34]: words that occur in a sequence in the document are collapsed into candidate phrases, and only phrases ended with nouns are extracted. Each extracted phrase is then scored using Equation 5:

$$PhraseScore(p) = \sum_{w_i \in p} S(w_i) \tag{5}$$

where p is a phrase, w_i is a word contained in p, and $S(w_i)$ is the word score from the ranking. Finally, the top n scored phrases are stemmed and extracted as the keyphrase for a document.

7. EXPERIMENTS

7.1 Dataset

Our model is evaluated on three publicly available datasets[2], Hulth2003, DUC2001, and SemEval2010. Each dataset contains documents of different length, we classify them into short, medium, and long document sets. The Hulth2003 dataset is the short document set, a collection of 2,000 abstracts of journal articles extracted from *Inspec* with approximately 100 to 150 words. DUC2001 contains 308 medium documents which are news articles with approximately 500 to 1,000 words each. The long document set, SemEval2010, collects 284 conference and workshop papers from the ACM Digital Library with approximately 2,000 to 10,000 words.

Each dataset contains a list of keyphrases assigned by either the authors or the readers, or both. These are considered the ground truth for our evaluation. However, a common issue with these datasets is that not all of the assigned keyphrases appear in the actual content of the texts. Thus the maximum recall score that one can obtain is less than 100 precent. For a fair and accurate evaluation, we remove all the documents in which not all the assigned keyphrases appear. After removal, the statistics for our evaluation dataset are presented in Table 1.

Table 1: Dataset Statistics

	TD	TP	TT	T/D	P/D
Hulth2003	500	3553	60,110	120.22	7.1
DUC2001	276	2233	252,524	915	8.1
SemEval2010	50	714	482,345	9647	14.3

TD: total number of the documents in the dataset. **TP:** total number of the assigned phrases. **TT:** total number of the tokens in the dataset (include punctuation marks, numbers, and symbols) . **T/D:**tokens per document. **P/D:** assigned keyphrases per document.

7.2 Evaluation Baseline and Methodology

Since we evaluate our work over three datasets with various document lengths, choosing a good baseline for our evaluation is not an easy task for a number of reasons. Firstly, many of the state-of-the-art algorithms are evaluated on their own chosen datasets (and often only on one dataset); thus the performance on different datasets remains unknown. Secondly, not all of the datasets that authors used are publicly available. Thirdly, the implementations of most of the state-of-the-art algorithms are not publicly available (apart from the very common ones, e.g. TF-IDF), and re-implementing these algorithms usually results in lower precision and recall than those claimed by the original authors due to different environments and configurations, such as using different stemmers or POS taggers.

Hasan and Ng [15] present an evaluation study, where they re-implemented some popular unsupervised extraction algorithms, such as TF-IDF [20], TextRank [27], SingleRank [34] and ExpandRank [34], and evaluated them on multiple datasets. They also made their re-implementations publicly available for evaluation and research purposes[3].

Therefore, in our evaluation, we use the re-implementations provided by Hasan and Ng, including TF-IDF and three other graph-based ranking algorithms: TextRank, SingleRank, and ExpandRank.

We use the *Precision*, *Recall*, and *F-measure* for evaluating the ranking algorithms. The *Precision* is defined as:

$$precision = \frac{the\ number\ of\ correctly\ matched}{total\ number\ of\ extracted} = \frac{TP}{TP+FP} \tag{6}$$

Recall is defined as:

$$recall = \frac{the\ number\ of\ correctly\ matched}{total\ number\ of\ assigned} = \frac{TP}{TP+FN} \tag{7}$$

F-measure is defined as:

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

[1]http://ml.nec-labs.com/senna/

[2]https://github.com/snkim/AutomaticKeyphraseExtraction

[3]Many thanks to Kazi Saidul Hasan and Vincent Ng for providing the re-implementation code.

Table 2: Comparison with baseline

	Algorithm	Parameter	Precision	Recall	F-score
Hulth2003	TF-IDF	W=5, N=10	35.29	48.33	40.79
	TextRank	W=5, T=0.3	15.31	23.78	18.63
	SingleRank	W=5, N=10	34.47	47.12	39.81
	ExpandRank	W=5, N=10	34.47	47.12	39.81
	WordAttractionRank	F = 1, N=10	**37.05**	**50.32**	**42.68**
DUC2001	TF-IDF	W=5, N=10	23.80	29.36	26.29
	TextRank	W=5, T=0.3	5.80	44.59	10.28
	SingleRank	W=5, N=10	21.67	26.72	23.93
	ExpandRank	N=10	21.88	27.0	24.17
	WordAttractionRank	F=1, N=10	**24.35**	30.09	**26.92**
SemEval2010	TF-IDF	W=5, N=15	7.07	7.42	7.24
	TextRank	W=5, T=0.3	2.26	57.00	4.35
	SingleRank	W=5, N=15	3.47	3.64	3.55
	ExpandRank	N=15	3.47	3.64	3.55
	WordAttractionRank	F=15, N=15	**13.5**	13.73	**13.61**

N: number of top ranked phrases **T:** percentage of vertices selected as keys that form phrases **W:** window size **F:** minimum word frequency

Table 3: Comparison with the state-of-the-art

	Algorithm	Precision	Recall	F-score
Hulth2003	Topic Clustering, Liu et al. [22]	35.0	**66.0**	**45.7**
	WordAttractionRank (F=1, N=10)	**37.1**	50.3	42.7
	WordAttractionRank (F=1, P=0.6)	32.3	65.2	43.2
DUC2001	ExpandRank, Wan et al. [34]	**28.8**	35.4	**31.7**
	WordAttractionRank (F=1, N=10)	24.4	30.1	26.9
	WordAttractionRank (F=1, N=15)	21.3	**39.5**	27.7
SemEval2010	Lopez et al. [24]	**27.2**	**27.8**	**27.5**
	WordAttractionRank (F=15, N=15)	13.5	13.7	13.6
	WordAttractionRank (F=15, N=20)	12.8	16.9	14.6

N: number of top ranked phrases **P:** percentage of top ranked phrases selected as keyphrases **F:** minimum word frequency

An assigned keyphrase matches an extracted phrase when they correspond to the same stem sequence. For example, *cleanup equipments* matches *cleanup equip*, but not *equip* or *equip cleanup*.

7.3 Experiment Results and Discussion

There are two hyper-parameters in our model. In the preprocessing stage, there is a hyper-parameter for minimum occurrence frequency of words; words that pass the threshold will be the vertices in the graph. Another hyper-parameter is in the post processing stage, indicating the number of top ranked phrases that are extracted as keyphrases. This can be set either to an absolute limit, or to the percentage of the total number of phrases required after ranking.

We conducted two experiments. The first experiment is to compare our results with the baseline. Here we used the re-implementation code provided by Hasan and Ng as baseline, and tested on the three refined datasets: Hulth2003, DUC2001, and SemEval2010. The re-implementation code does not contain a parser, so we had to POS tag all documents using the Stanford parser before the experiment. For each run, we extracted the top 10 ranked phrases as the keyphrases (apart from TextRank in which only the percentage of the vertices of the graph can be selected), and then both ground truth (assigned keyphrases) and extracted phrases were stemmed before comparison. The scores obtained for the four baseline algorithms are very close to the evaluation results presented by Hasan and Ng, but not identical. This is because we cleaned our dataset ensuring that all the assigned keyphrases appear in the documents. When extracting keyphrases, there are many factors affecting the overall performance, such as the preprocessing steps, stemming, tokenising, POS tagging, and phrase formation. In our experiment, we have minimised these factors by using the same tokeniser, POS tagger, and phrase formation techniques (apart from TextRank which uses different phrase formation technique).

The experimental results are presented in Table 2. Our model produced superior performance over other baseline algorithms uniformly across different datasets. We found that the key factor that differentiates the performance is the weighting scheme comparing the three other graph-based ranking algorithms, TextRank, SingleRank, and ExpandRank. TextRank uses an unweighted PageRank algorithm; SingleRank uses co-occurrence frequencies as the weights; and ExpandRank uses the co-occurrence frequencies calculated from both the document itself and the neighbour documents. We further evaluated these weighting schemes on fetching individual words using the same preprocessor, stemmer, tokeniser and POS tagger. Our results show that our weighting scheme outperforms the unweighted graph by about 4-5% on F-score, and outperforms the co-occurrence weighting scheme by about 2-3% on F-score.

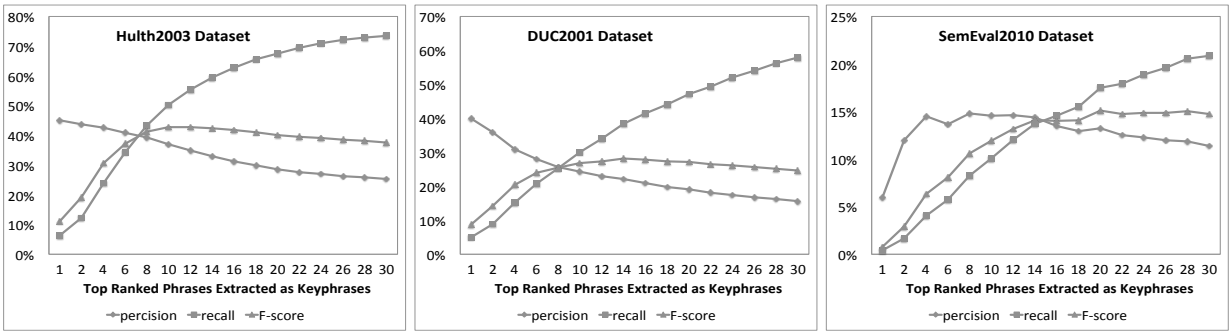

Figure 3: Precision, recall and F-score curves for three datasets.

In the second experiment, we compare our results with the state-of-the-art algorithms. As mentioned before, many algorithms were usually evaluated only on a single purposely chosen dataset; thus, there are three state-of-the-art algorithms on our chosen datasets. The performance data of each algorithm were collected from a recent survey paper by Hasan and Ng [16], and only the best results from the studies are shown. The experiment results are presented in Table 3.

Overall, our model performed very close to the state-of-the-art algorithms on short and medium length document sets. However, the performance of state-of-the-art algorithms is often tuned to a specific dataset. Our model, on the other hand, is generic and corpus-independent. Comparing with the state-of-the-art algorithms, our model is penalised heavily by the trade-off between precision and recall, whereas the state-of-the-art performance offers better recall without losing much on precision. Figure 3 shows the precision, recall, and F-score curves of our model on the three datasets.

Clearly, the overall performance of all ranking algorithms drops with an increase in document length, and our model is no exception. On the long document set, we are still far from the state-of-the-art performance because we do not resort to the aid of heuristics. Note that the model presented by Lopez et al. [24] is a supervised machine learning approach. The best performed unsupervised approach on the same dataset is presented by El-Beltagy and Rafea [9], which shows an F-score of 25.2. Similarly, El-Beltagy and Rafea tune the approach on the given dataset using heuristic rules, such as setting a threshold that looks for the position at which the first keyphrase appears, and then discarding all words appearing before that position.

8. CONCLUSION AND FUTURE WORK

In this paper, we have introduced a weighting scheme for unsupervised graph-based keyphrase extraction. The weighting scheme calculates the attraction scores we defined between words using the background knowledge provided by word embedding vectors. We evaluated our model using datasets of various document lengths and demonstrated effectiveness of the model that outperforms all the chosen baseline algorithms. Although our algorithm is not able to outperform the state-of-the-art algorithms, we consider that the model can be further improved by training the embeddings over the target dataset using pre-trained embeddings as initial values. In this way, the embeddings can capture not only the general knowledge, but also specific domain knowledge about the dataset. Therefore, our future work will focus on investigating and improving the background knowledge that word embeddings can capture, in order to further improve our model.

9. ACKNOWLEDGMENT

This research was funded by the Australian Postgraduate Awards Scholarship, Safety Top-Up Scholarship by The University of Western Australia, and linkage grant LP110100050 from the Australian Research Council.

10. REFERENCES

[1] Y. Bengio. Neural net language models. *Scholarpedia*, 3(1):3881, 2008.

[2] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003.

[3] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1):107–117, 1998.

[6] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[7] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[8] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[9] S. R. El-Beltagy and A. Rafea. Kp-miner: Participation in semeval-2. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 190–193. Association for Computational Linguistics, 2010.

[10] G. Ercan and I. Cicekli. Using lexical chains for keyword extraction. *Information Processing & Management*, 43(6):1705–1714, 2007.

[11] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin,

and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *PROC. SIXTEENTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE*, pages 668–673. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.

[12] M. Grineva, M. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *Proceedings of the 18th international conference on World wide web*, pages 661–670. ACM, 2009.

[13] M. A. K. Halliday and R. Hasan. *Cohesion in English*. Routledge, 2014.

[14] Z. S. Harris. Distributional structure. *Word*, 1954.

[15] K. S. Hasan and V. Ng. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 365–373. Association for Computational Linguistics, 2010.

[16] K. S. Hasan and V. Ng. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1262–1273, 2014.

[17] G. E. Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA, 1986.

[18] G. E. Hinton. Connectionist learning procedures. *Artificial intelligence*, 40(1):185–234, 1989.

[19] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 216–223. Association for Computational Linguistics, 2003.

[20] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

[21] D. Klein and C. D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[22] Z. Liu, W. Huang, Y. Zheng, and M. Sun. Automatic keyphrase extraction via topic decomposition. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 366–376. Association for Computational Linguistics, 2010.

[23] Z. Liu, P. Li, Y. Zheng, and M. Sun. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 257–266. Association for Computational Linguistics, 2009.

[24] P. Lopez and L. Romary. Humb: Automatic key term extraction from scientific articles in grobid. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 248–251. Association for Computational Linguistics, 2010.

[25] K. Lund and C. Burgess. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208, 1996.

[26] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(01):157–169, 2004.

[27] R. Mihalcea and P. Tarau. TextRank: Bringing Order into Texts. In *Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, 2004.

[28] T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. Citeseer, 2013.

[29] Y. Ohsawa, N. E. Benson, and M. Yachida. Keygraph: Automatic indexing by co-occurrence graph based on building construction metaphor. In *Research and Technology Advances in Digital Libraries, 1998. ADL 98. Proceedings. IEEE International Forum on*, pages 12–18. IEEE, 1998.

[30] M. Stubbs. Two quantitative methods of studying phraseology in english. *International Journal of Corpus Linguistics*, 7(2):215–244, 2003.

[31] T. Tomokiyo and M. Hurst. A language model approach to keyphrase extraction. In *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pages 33–40. Association for Computational Linguistics, 2003.

[32] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[33] P. D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.

[34] X. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In *AAAI*, volume 8, pages 855–860, 2008.

[35] J. Wang, J. Liu, and C. Wang. Keyword extraction based on pagerank. In *Advances in Knowledge Discovery and Data Mining*, pages 857–864. Springer, 2007.

[36] W. Wong, W. Liu, and M. Bennamoun. Determining termhood for learning domain ontologies using domain prevalence and tendency. In *Proceedings of the sixth Australasian conference on Data mining and analytics-Volume 70*, pages 47–54. Australian Computer Society, Inc., 2007.

[37] W. Wong, W. Liu, and M. Bennamoun. Determining the unithood of word sequences using mutual information and independence measure. *arXiv preprint arXiv:0810.0156*, 2008.

[38] W. Xing and A. Ghorbani. Weighted pagerank algorithm. In *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*, pages 305–314. IEEE, 2004.

[39] W.-t. Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *Proceedings of the 15th international conference on World Wide Web*, pages 213–222. ACM, 2006.