ACL vs NIPS: Keyword Phrase Extraction using GloVe Word Embeddings

Mayuri Kumari
University of Illinois at Chicago
mkumar29@uic.edu

Rashmi Arora
University of Illinois at Chicago
rarora20@uic.edu

ABSTRACT

The document space is growing at a humongous rate and volume everyday. This poses a need for generating keywords, accurately and efficiently, for a faster and easier analysis of the huge amount of textual information available . In this project, we study keyphrase extraction from research papers for two different conferences (ACL and NIPS). We propose a biased TextRank algorithm based on the notion that certain sections of a research paper hold more important information than others. We also incorporate word embeddings from the GloVe model to generate more meaningful association between words. This is then combined with Inverse Distance Weighting scheme for a chosen window size over the text. The model outperforms the basic TextRank algorithm and is shown to improve the Mean Reciprocal Rank by 10-15% for the given dataset.

1 INTRODUCTION

Keyphrase extraction is the task of identifying terms that best describe the subject of a document.[1] Keyphrases or just keywords are the terms that represent the most relevant information contained in the document. Extracting keywords is one of the most important tasks while working with text data in the domain of Text Mining, Information Retrieval and Natural Language Processing. It helps in query refinement, document indexing, text summarization, identifying potential readers, recommending documents to them and analyzing research trends over time.

In order to find out the documents related to our interests in a document space that is growing at an enormous speed and scale, we need to analyze large quantities of text data. This can be made easier with Keyphrase Extraction which can provide us with a high level description or main theme of the document and would thus, help us to organize documents and retrieve them easily based on these important keywords.

There are mainly two approaches to perform Keyphrase Extraction in text documents - Supervised and Unsupervised approach. In the supervised learning method, keyphrase extraction is generally implemented as a binary classification problem[6] such that if a candidate phrase is classified as positive, it is considered as keyphrase and if classified as negative, it is counted as non-keyphrase. The unsupervised methods, on the other hand, are domain-independent but require huge amount of data to produce any significant results. These are mostly based on TF-IDF, clustering, and graph-based ranking[8].

Existing approaches to supervised learning for keyphrase extraction though produce better results than unsupervised methods, but these require labeled training data which is difficult also to generate and also generalize very poorly outside the domain of the training data. With recent advancements in deep learning techniques, words can be easily represented as dense real-valued vectors, also known as *word embeddings*. These vector representations of words are supposed to preserve the semantic and syntactic similarities between words and have been shown to equal or outperform other models like Latent Semantic Analysis, etc. Two of the most popular models for training word embeddings are Word2Vec[9]and GloVe[10].

In this project, we implement an unsupervised graph-based method for Keyphrase Extraction from research papers where the words are treated as nodes and the association between the words is depicted by the edges. We use parameterized edge weights from similarity between the 50-dimensional word embeddings from GloVe model, weights corresponding to different sections and inverse distance weights (weights assigned to neighboring words in decreasing order with increasing distance).

2 RELATED WORK

2.1 Supervised Approach

Uzun et al., 2005 used Naı̈ve Bayesian classifier with the assumption that the keyword features are normally distributed and independent. It made use of features such as TF-IDF score, distance of the word from the beginning of the text, paragraph and the sentence to identify keywords in the text[11].

KEA (Keyphrase extraction algorithm) was developed by Frank et al., 1999 in which a classifier is build based on the Bayes theorem from training documents, and then is used to extract keyphrases from new documents. Two features are utilized: TF-IDF and first occurrence of the term[5].

Bao Hong et al., 2012 used Support Vector Machines on features obtained from word frequency, part of speech, syntactical function of words, position appeared & word's morphology. Different weights were assigned based on these different features.

2.2 Unsupervised Approach

Liu et al., 2009 adopt a clustering-based approach, KeyCluster, that clusters semantically similar candidates using Wikipedia and co-occurrence-based statistics. The underlying hypothesis is that each of these clusters corresponds to a topic covered in the document, and selecting the candidates close to the centroid of each cluster as keyphrases ensures that the resulting set of keyphrases covers all the topics of the document[7].

Another graph-based and unsupervised approach to keyphrase extraction incorporates information from all positions of a word's occurrences into a biased PageRank to extract keyphrases[4] as proposed by Florescu and Caragea, 2017.

Liu et al., 2010 propose Topical Page Rank which runs TextRank multiple times for a document, once for each of its topics induced by a Latent Dirichlet Allocation. The final score of a candidate is computed as the sum of its scores for each of the topics, weighted by the probability of that topic in that document.

Caragea et al., 2014 proposed an algorithm called CiteTextRank,an unsupervised, graph-based approach which incorporates evidence from multiple sources (citation contexts as well as document content) in a flexible way to extract keyphrases[3].

Bennani-Smires et al., 2018 use sentence embeddings to represent (embed), both the candidate phrases and the document itself in the same high-dimensional vector space and rank the candidate phrases to select the output keyphrases[2].

3 PRE-PROCESSING

Since the data for ACL was in pdf format, we used Linux's 'pdftotext' command to convert the pdf files into plain text format. Further, we analyzed previous years ACL and NIPS Research papers and found that in most of the papers, sections 'Title', 'Abstract' and 'Conclusion' which contains more important information about the paper were present. Since each section holds different importance about the main subject or topic of the paper, we extracted each of these sections 'Title', 'Abstract', 'Conclusion' and 'References' text from the whole paper text and rest we considered as body text of the paper and preprocessing was done for each of the sections as well as body text separately.

Data cleaning was performed on each section text separately. First sentence of the Abstract and Conclusion sections has more relevant words about the subject than the rest sentences in that section. We used this varied importance of sentences about the paper topic as one of the key factors while assigning weights to different words belonging to sentences in different position and section of the paper text. We used NLTK sentence tokenizer to tokenize each section text.

We performed data preprocessing on each paper text which includes word tokenization, stop words removal, check for alphanumeric and not a digit, POS tagging, stemming and removal of very short word. Porter Stemmer was used for POS tagging and only nouns and adjectives corresponding to POS tags 'NN', 'NNS', 'NNP', 'NNPS' and 'JJ' were used for the computation of keywords extraction.

4 PROPOSED MODEL

Each sentence words belonging to different position in different sections were assigned different weights depending on their relevance importance of the main subject of the paper. All the sentences with assigned weights from each of the sections were combined to prepare the weighted word graph for text rank computation. Section and Position based weight assignments used in our computation are shown in Table 1.

Words belonging to each sentence were assigned the sentence weights and these weights were used while preparing the word graph and assigning initial score to words in the biased text rank computation. We incorporated the term frequency of each word in the combined text by adding the weight of the word in the sentence from were it was read. Word weights were the combination of weight of the word from different sections and positions of the text. This also incorporated the weighted term frequency of the word in the text. We used these word weights for assigning the initial score of the word in the biased text rank computation.

Table 1: Weight assignment to different sections

Section	Weight
Title	1
Abstract First Sentence	0.85
Abstract Rest	0.75
Conclusion First Sentence	0.85
Conclusion Rest	0.75
Remaining Text	0.7

For the creation of the word graph, we computed the weights between the words of the sentence by the sentence weight and how far the words are positioned in that sentence. Words that were next to each other were assigned the weight of the sentence as their edge weight. Edge between the words that were one or two positions apart from each other were assigned reduced sentence weight. Edge weight of the words that were one position apart from each other was assigned 3/4th of the sentence weight and edge weight of the words that were two positions apart from each other was assigned half of the sentence weight. We kept the window size as 3 for distance-wise edge weight computation of the words in the word graph.

We incorporated word embedding, vector space models of word semantics for our computation of the edge weight of the words in the paper text. We used Glove: Global Vectors for Word Representation pre-trained word vectors for this. Word embedding is one of the most popular representation of document or corpus vocabulary. It captures context of a word in a document or corpus, semantic and syntactic similarity, relation with other words. It reconstruct linguistic contexts of words and produces a vector space, of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space.Word vectors are positioned in the vector space such that words that share common contexts in the corpus are in close proximity to one another in the space.

The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in the corpus. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. We leveraged the capability of word embedding of providing semantic and syntactic relation between words in the assignment of the edge weight between words for the creation of weighted word graph. We calculated the words similarity by using cosine similarity on the word vectors obtained from GloVe pre-trained word vectors. This cosine similarity value was added in the edge weight of the words calculated in previous step using weights of the word in different sections and positions in the document.

We used Biased Text Rank for our computation of calculating the scores of each unique word in the document. Text Rank is graph-based algorithm which is a way of deciding on the importance of a vertex within a graph, by considering global information recursively computed from the entire graph, rather than relying only on local vertex-specific information. The basic idea implemented by a graph-based Text ranking model is that of 'voting' or 'recommendation'. When one word is present next to or near next to another one that is links to another word, it is basically casting a vote for that other word. The higher the number of votes that are cast for a word, the higher the importance of the word. Moreover, the importance of the word casting the vote determines how important the vote itself is, and this information is also considered by the text ranking model. Hence, the score associated with a word is determined based on the votes that are cast for it, and the score of the words casting these votes.

Consider s(v_i) is the initial score of the word, w_{ji} is the edge weight of word i with its adjacent words j in consideration and w_{jk} is the edge weight between i's adjacent word j and their adjacent word k. We used below mentioned formula of Text Rank to compute the scores of the words using weighted word graph prepared in previous step:

$$s(v_i) = \alpha \cdot p(v_i) + (1-\alpha) \sum_{v_j \in Adj(v_i)} \frac{w_{ji}}{\sum\limits_{v_k \in Adj(v_j)} w_{jk}} s(v_j)$$

where α is a damping factor (α = 0.15) and p = [1/n, ..., 1/n] and n is number of words in the word graph.

State-of-the-art text rank algorithm used initial scores of each word as 1/n but since we have already determined weight or importance of each unique word in the document based on its presence with respect to section and position and its frequency in different sections, we used this word importance weight as the initial score in the text rank algorithm. This approach outperformed the state-of-the-art algorithm used for key phrase extraction.

Formation of n-grams and its score computation: Since the keywords for the paper are not only single word but combination of syntactic related words, we formed n-grams and computed its score by adding the score of the words constituting the n-gram. We then sorted this dictionary of n-gram and unigram scores based on the score value. We retrieved top twenty n-grams from this sorted dictionary of n-gram scores which formed the keywords for each research paper.

5 EXPERIMENTS

Dataset and Evaluation Measure

The dataset comprised of ACL Research papers from ACL Anthology Reference Corpus[1]. It had 9849 ACL research papers all in pdf format. For NIPS papers, we used the NIPS dataset available at Kaggle[2]. This dataset includes 7241 NIPS papers ranging from year 1987 to 2016 conference.

The evaluation measure used was Mean Reciprocal Rank (MRR) which can be computed as,

$$MRR = \frac{1}{|D|} \sum_{d=1}^{|D|} \frac{1}{r_d}$$

where r_d is rank at which first correct prediction for d was found.

[1]http://acl-arc.comp.nus.edu.sg/

[2]https://www.kaggle.com/benhamner/nips-papers

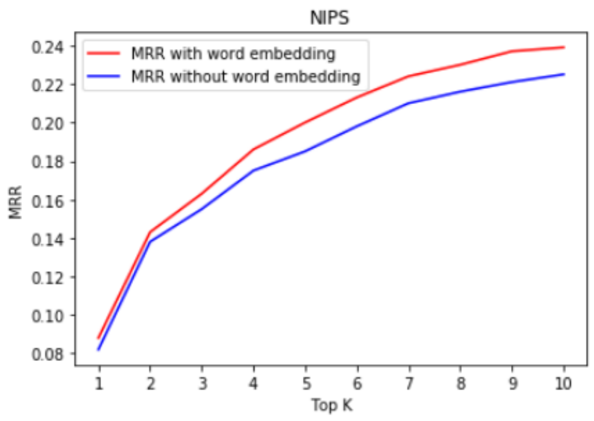

Figure 1: Comparison of MRR with word embeddings and without word-embeddings for NIPS papers

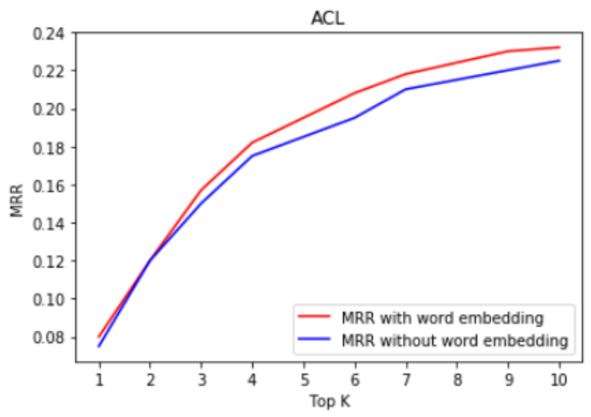

Figure 2: Comparison of MRR with word embeddings and without word-embeddings for ACL papers

Gold standard keywords for NIPS papers were obtained from NIPS official website[3].

Gold standard key words of ACL papers from ACL Anthology Reference Corpus[4]. We used these gold standard keywords for evaluating our approach of key phrase extraction.

Results and Observation

The results for MRR with and without word embeddings for NIPS and ACL papers are shown in Figure 1 and Figure 2 respectively. We evaluated top-K predicted keywords with K ranging from 1 to 10 against the gold standard keywords for both NIPS and ACL with and without word-embeddings. The maximum MRR value of 0.239 was achieved at K=10 for NIPS papers with word embeddings. The corresponding maximum value for ACL was 0.232, with word-embeddings. Incorporating section weights and inverse distance weighting scheme helped improve the MRR by 13.8% over the basic TextRank. Also, it can be inferred that word-embeddings helped

[3]https://nips.cc/

[4]http://pars.ie/lr/acl-rd-tec-terminology/_acl_arc_comp/annotation

form better word associations and thus, a better MRR value.

Challenges

The major challenge that we faced was processing the huge amount of data. We processed and ran our algorithm on more than 17,000 files available (9849 ACL research papers and 7241 NIPS research papers) along with 332,470 word embeddings (50-dimensional) which was time-consuming.

Other major challenge was hyper-parameter tuning the section weights. We tried a few different combinations of feasible weight assignments such that no section is too undervalued or too overvalued. Significant improvement in results was achieved using the section weights as mentioned in Table 1.

6 CONCLUSION

In this project, we proposed an unsupervised keyphrase extraction algorithm for scientific papers and compared the results for two different conferences, ACL and NIPS. We implemented a biased TextRank algorithm using parameterized weights instead of fixed weights. Since, certain sections of a research paper hold more important information than others, we used different weights for different sections in the research papers. We also incorporated word embeddings from the GloVe model to generate more meaningful association between words and combined it with Inverse Distance Weighting scheme for a chosen window size of 3 over the text. The model outperforms the basic TextRank algorithm and is shown to improve the Mean Reciprocal Rank by 10-15% for the given dataset. Also, the word-embeddings from GloVe helped form better word associations and thus, helped achieve improved MRR values.

7 FUTURE WORK

We limited our study in the current project to using word embeddings from GloVe model only. In future, we plan to experiment with word embeddings from other models like Word2Vec and fastText and compare these different models.

Also, we plan to test our model on domains outside computer science such as Ecology, Chemistry, Social Science, etc.

ACKNOWLEDGMENTS

We would like to thank Prof. Cornelia Caragea for encouraging and motivating us during the project. She helped in clarifying all our doubts while incorporating word embeddings in our algorithm and always guided us towards the right direction.

We would also like to thank TA George Maratos for providing help in better understanding the PageRank algorithm which thus, motivated us to experiment with the state of the art algorithm and introduce modifications to it.

REFERENCES

[1] Slobodan Beliga, Ana MeÅątroviÄĞ, and Sanda Martincic-Ipsic. 2015. An Overview of Graph-Based Keyword Extraction Methods and Approaches. *Journal of Information and Organizational Sciences* 39 (07 2015), 1–20.

[2] Kamil Bennani-Smires, Claudiu Musat, Andreea Hossmann, Michael Baeriswyl, and Martin Jaggi. 2018. EmbedRank: Unsupervised Keyphrase Extraction using Sentence Embeddings. (01 2018).

[3] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-Enhanced Keyphrase Extraction from Research Papers: A Supervised Approach. 1435–1446. https://doi.org/10.3115/v1/D14-1150

[4] Corina Florescu and Cornelia Caragea. 2017. A Position-Biased PageRank Algorithm for Keyphrase Extraction.

[5] Eibe Frank, Gordon W. Paynter, Ian Witten, Carl Gutwin, and Craig Nevill-Manning. 1999. Domain-Specific Keyphrase Extraction. (07 1999).

[6] Kazi Saidul Hasan and Vincent Ng. 2014. 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)*. Association for Computational Linguistics, Baltimore, Maryland, 1262–1273. http://www.aclweb.org/anthology/P14-1119

[7] Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to Find Exemplar Terms for Keyphrase Extraction. In *EMNLP*.

[8] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing Order into Text.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems* 26 (10 2013).

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[11] Yasin Uzun. 2005. Keyword Extraction Using Naive Bayes. (01 2005).