The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

Exploiting Textual and Citation Information to Identify and Summarize Influential Publications

Mohamed A. Zahran
Purdue University
Department of Computer Science
West Lafayette, Indiana, USA
mzahran@purdue.edu

Amr Ebaid
Purdue University
Department of Computer Science
West Lafayette, Indiana, USA
aebaid@purdue.edu

Abstract

Given a group of publications, we investigate the problem of identifying the papers with the most impact on others. We refer to these papers as *influential* in the sense that they introduce new concepts and language that will affect how future articles are written. In this paper we propose weighted PageRank algorithm that uses textual information from articles and information from citation graph to rank the impact of publications, then we automatically summarize these publications and extract important keywords. We show that using our algorithm outperforms default citation-based techniques in ranking influential papers (those which won best paper award) with no less than 2% in F1-score and NDCG. We also show that our algorithm outperforms previous graph-based keyword extraction techniques with no less than 1.5% in F1-score.

Introduction

There exists a significant body of work in citation graph analysis and modeling. Popular ranking algorithms as PageRank and HITS were used in the context of citation graphs to give different types of rankings as venue ranking, author ranking and publication ranking. While previous work shows that the default PageRank works for publication ranking, textual information from publications are not used in this line of work to influence the ranking. In this paper we use textual information as well as graph information to pick the top rated influential publications and determine the type of new language and vocabulary these publications influenced other articles and publications. After picking those influential publications we provide a summary for each paper. The summary is basically to extract the top important sentences, then for each of these sentences we extract the most important words as keywords.

More formally: Given a group of publications $P = \{p_1, p_2...p_n\}$ where p_i is a publication. A publication is a set of sentences $p_i = \{s_1, s_2...s_m\}$ where s_i is a sentence. A sentence is a set of words $s_i = \{w_1, w_2...w_t\}$ where w_i is a word. It is required to do these three tasks:

1. **Publication Ranking**: Given P, we identify a subset $P^* \subseteq P$ having the k_p highest publication scores in P.
2. **Extractive Document Summarization**: Given a paper p we identify a subset $S^* \subseteq p$ having the k_s highest scores in p.
3. **Keyword Extraction**: Given a sentence s we identify a subset $W^* \subseteq s$ having the k_w highest word scores in s.

Related Work

Our work is divided into three tasks as shown in the previous section. We will divide the related work into three sections as well.

Publication Ranking

Paper ranking can be done by mere citation count such that papers are ordered descendingly by the frequency of their corresponding citations. A better way is to view papers as nodes, citations as edges and apply PageRank to calculate the rank of each paper. While this idea works, this method does not accommodate for the relative impact of publications or the impact of venues. (Sun and Giles 2007) proposed using a Popularity Factor (PF) to reflect the influence of a publication's venue. The type of venue whether it is a journal, conference or a workshop is transparent to the popularity factor.

$$PF(v,t) = \frac{n_v}{N} * \sum_{i \epsilon P} \frac{PF(i,t) \times w(i)}{N(i)} \quad (1)$$

Where $PF(v,t)$ is the popularity factor for venue v in a given year t ranging from 0 to 1. n_v is the number of papers published in venue v in year t, N is the total number of publications in all venues in year t, P is the set of venues which cite this venue v in year t, $w(i)$ is the frequency that venue i cites venue v and $N(i)$ is the total number of citations by venue i.

Then using the popularity factor, they calculate the ranking score of a paper as follows:

$$R(d_T) = PF(v_{d_T}) + \sum_{t>T, d_t \epsilon D} \frac{R(d_t)}{N(d_t)} \quad (2)$$

Where $R(d_T)$ is the score of paper d at time T, $PF(v_{d_T})$ is the popularity factor of venue publishing the paper d_T. $N(d_t)$ is the number of references in paper d_t and D is the set of papers citing d_T

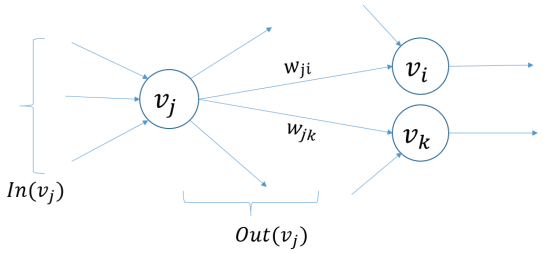

Figure 1: Simple Citation Graph Example

On the other hand, another line of work proposed using solely textual information to identify influential papers. Using an NLP approach, (Gerrish and Blei 2010) suggested that an influential article will affect how future articles are written and that this effect can be detected by examining the way publications corpus statistics change over time. Their hypothesis is that an article's influence on the future is captured by how the language of its field changes subsequent to its publication. Their proposed model is based on dynamic topic modeling.

(Ramage, Manning, and McFarland 2010) give ranking to institutions by analyzing how futuristic their research is. They perform topic modeling on the PhD dissertations abstracts and check if these topics match the future publications' topics.

Extractive Document Summarization

Focusing on graph based approaches for extractive summarization, (Mihalcea and Tarau 2004) proposed generating a graph with sentences as nodes and edges between these nodes represent a similarity relation between the sentences. Sentences appearing within a certain window are considered neighbors. Similarity is measured as a function of their content overlap. For the sentence $S_i = w_1^i, w_1^i, w_2^i ... w_{m_i}^i$ and $S_j = w_1^j, w_1^j, w_2^j ... w_{m_j}^j$, the content overlap based similarity is:

$$Sim(S_i, S_j) = \frac{|\{w_k | w_k \epsilon S_i \& w_k \epsilon S_j\}|}{log(|S_i|) + log(|S_j|)} \quad (3)$$

The intuition is that adding similarity as weights on the graph edges between two sentences can be seen as a process of recommendation. A sentence that addresses certain concepts in a text, gives the reader a recommendation to refer to other sentences in the text that address the same concepts, and therefore a link can be drawn between any two such sentences that share common content. Then using weighted PageRank (equation 4) the extractive summary will be the top rated k_s sentences.

$$WPR(v_i) = (1-d) + d * \sum_{v_j \epsilon In(v_i)} \frac{w_{ji}}{\sum_{v_k \epsilon Out(v_j)} w_{jk}} WPR(v_j) \quad (4)$$

From Figure 1 we can see that $In(v_j)$ is the set of nodes with incoming edges into v_j, while $Out(v_j)$ is the set of nodes v_j points towards them. d is a damping factor that can be set between 0 and 1 (usually 0.85).

Keyword Extraction

Extracting important words from a paragraph is known in the NLP community as keywords extraction problem. (Mihalcea and Tarau 2004) proposed to use the input paragraph to construct a graph between words. The graph can be directed or undirected. Words appearing within a certain window W will have an edge joining them, then they calculate the PageRank score for words and pick the k words with maximum score. The edges between words can be either weighted or unweighted. They used unweighted undirected graphs in their experiments. They also experimented with various syntactic filters and reported the best results by ruling out all words except for nouns and adjectives.

In this paper, we combine a graph based approach using citation graph together with textual clues to identify and rank influential papers, then using graph based approaches we summarize these papers and extract the most important keywords.

Methodology

Our work here involves three tasks, publication ranking, extractive document summarization, keyword extraction. We propose using graph-based techniques to solve all three tasks. As discussed we can view all of the three tasks as ranking tasks. The idea is to convert the ranking problem into a graph, such that nodes are the items to rank, and edges are the relationships between them. Then, using edge weighted PageRank we can rank these items and pick the top rated ones.

To apply this idea we need: First, to use the appropriate relation metric between items. Each of the three tasks will define its own function to assign a relationship score for each two connected items. Second, We need to modify the weighted PageRank (equation 4). For PageRank to converge, $\sum_{v_j \epsilon Out(v_i)} w_{ij} = 1$, i.e. the summation of weights of all the outgoing edges for any node must be 1. This can be problematic in our tasks which depends on the relationship score assigned to these edges. For example, considering Figure 1, let $Out(v_j) = v_i, v_k$ and assume $w_{ji} = 0.1$ and $w_{jk} = 0.025$. In order to satisfy the above condition, we should normalize these weights to be $w_{ji} = \frac{0.1}{0.1+0.025} = 0.8$ and $w_{jk} = \frac{0.025}{0.1+0.025} = 0.2$. We can see that the weights significantly increased. If these weights represent a similarity measure between two nodes, then the similarity score jumped from 0.1 (very low similarity) to 0.8 (very high similarity). A simple work around is to do the normalization only if the $\sum_{v_j \epsilon Out(v_i)} w_{ij} > 1$. As long as the summation of the out weights are less than one (as in our example) no normalization is required and PageRank will run and converge. In our experiments, we follow this simple trick which improved our results. Next we present the methodology used in each task.

Publication Ranking

The main idea is that all citations should not be treated equally. Previous work treats the citations to other publi-

cations equally, which fails to account for the relative importance of one citation over another. The importance of a citation is directly proportional to the similarity of context and ideas between the two papers.

For any research article, it make sense that its dependency (represented in citations) on other publications is not uniform; For example, a paper can cite another publication for just using a dataset, for a background and related work or as self-citation. These types of citations should not get high weights. On the other hand, a citation can be an extension of some previous work, or enhancement/comparison of certain aspects... etc. These types of citations should receive higher weights because the cited articles have higher influence on the paper at hand. The magnitude of this influence is somehow subjective. However, if we have two papers v_j and v_i, and the topics discussed in v_j are very different from those discussed in v_i, then we can assume that it is highly unlikely that v_j cites v_i or vice versa. The reason is that for v_j to cite v_i they should have some common topics. The more similar the topics of v_j are to those of v_i, the more impact v_i has on v_j.

Using NLP techniques, we can give a score to the similarity between two papers. The higher the score, the more important the citation is. Similarity scores are used as weights on the edges of the citation network. More precisely, consider Figure 1 where v_j cites v_i. The weight on the citation edge $w_{ji} = sim(v_j, v_i)$ where $sim(.)$ is a similarity function that receives the textual information for both papers and returns a similarity score between them.

We investigated two NLP approaches to assess the similarity score between two documents:

Topic Modeling Using topic modeling techniques as LDA (Blei, Ng, and Jordan 2003), we can analyze the topics of the paper at hand, and compare these topics to the cited publication. The weight of such citation is directly proportional to the similarity between the topics of the two papers. In our experiments, we pooled all the publications together to identify all topics using gensim [1]. The number of topics we used is 100. The similarity between two papers will be by comparing how close the topics of the two papers are to each other.

Document Embedding Instead of just analyzing the topics of the papers, we can go a bit deeper by considering the context of each paper, such that two papers with similar contextual information should receive a high score.

(Mikolov et al. 2013) proposed two techniques for building word representations in vector space using log linear models; continuous bag of words (CBOW) and Skipgram models. An extension to this work is made to embed a whole document not just words. (Le and Mikolov 2014) proposed to train document vectors the same way as words. The document can be thought of as another word. It acts as a memory that remembers what is missing from the current context, and as the sliding window moves over the corpus, words in the window change but the document is still present, such that the document vector is shared across all contexts generated from the same document but not across documents. The words vectors on the other hand are shared across all documents.

In our experiments, we pooled all the publications together to train the document vector of each paper. The vector size in our experiments is 300 and the sliding window is 10. We also filtered out all words with frequency less than 3, then trained the model using gensim. The similarity between two papers will be cosine the angle between their corresponding document vectors: $Sim(paper_i, paper_j) = Max\{\epsilon, Cos(v_i, v_j)\}$ where ϵ is a very small positive number. The max function is here to reject negative cosine scores and return ϵ instead.

Extractive Document Summarization

Following the work of (Mihalcea and Tarau 2004), for a given document, we will represent sentences as nodes, sentences within a certain window are considered neighbors, with edges weighted by the similarity score between the connected sentences. But, rather than the similarity measure they proposed in equation 3, we will use trained word embeddings [2] to get a vector representation for a sentence using Min/Max/Sum pooling. Given a sentence, break it into words, and use the individual vectors of the words making up the sentences to represent the sentence as a vector as well. Pool the minimum components of all word vectors, then concatenate it with the maximum components of all word vectors, concatenated with the sum for all word vectors. Let the sentence $S = w_1, w_2...w_t$ Let v_i be the vector corresponding to word w_i and $v_i(x)$ is the x^{th} dimension of the vector, and let v_s be the sentence vector.

$$v_{min} = [Min(v_i(x) \forall i \epsilon [1..t], \forall x \epsilon [1..|v_i|])]$$

$$v_{max} = [Max(v_i(x) \forall i \epsilon [1..t], \forall x \epsilon [1..|v_i|])]$$

$$v_{sum} = [Sum(v_i(x) \forall i \epsilon [1..t], \forall x \epsilon [1..|v_i|])]$$

Then $v_s = v_{min}|v_{max}|v_{sum}$ where $|$ is the concatenation operator. The main reason to use this representation over the document vector is that sentences have considerably fewer number of words which will make the document vector not very accurate and require a lot of training data, which suggests that combining individual word vectors is a better option. Also we can find pretrained word vectors on a massive amount of data as word2vec which is more likely to give better vector representation for words and in turn better sentence vectors for short sentences.

Now the similarity between two sentences s_i, s_j will be cosine the angle between the two vectors: $Sim(s_i, s_j) = Max\{\epsilon, Cos(v_i, v_j)\}$ where ϵ is a very small positive number. The max function is here to reject negative cosine scores and return ϵ instead.

Keyword Extraction

Following the work of (Mihalcea and Tarau 2004), We represent words as nodes, and the edges carry weights representing the similarity between connected words. We propose using weighted edges between words such that a weight on

[1] https://radimrehurek.com/gensim/

[2] https://code.google.com/archive/p/word2vec/

the edge between two words represent a similarity score between these two words. Then, we use PageRank to get the top k words to be the extracted keywords. We experimented with these word to word similarity measures:

1. Word Vector Embeddings Similarity: Using the word vectors discussed, then calculate the cosine similarity between the two word vectors.
2. Path Similarity: The score denoting how similar two word senses are, based on the shortest path that connects the senses in the is-a (hypernym/hypnoym) taxonomy as in WordNet.
3. The Leacock & Chodorow (Lch) Measure (Leacock and Chodorow 1998):

$$Sim_{lch}(c_1, c_2) = -\log \frac{len(c_1, c_2)}{2 * D}$$

Where c_1 and c_2 are the two concepts corresponding to the two words in question. len is the length of the shortest path between the two concepts c_1 and c_2 using node counting, and D is the maximum depth of the taxonomy.

4. The Wu and Palmer (Wup) Measure (Wu and Palmer 1994):

$$Sim_{wup}(c_1, c_2) = \frac{2 * depth(LCS(c_1, c_2))}{depth(c_1) + depth(c_2)}$$

Where LCS is the least common subsumer of the two concepts.

5. The Lin Measure (Lin 1998):

$$Sim_{lin}(c_1, c_2) = \frac{2 * IC(LCS(c_1, c_2))}{IC(c_1) + IC(c_2)}$$

Where IC is the information content, such that the information content of a concept c equals to $IC(c) = -\log P(c)$ where $P(c)$ is the probability of encountering c in a large corpus.

Algorithm/Data Description

Algorithm

In this section we present our modified weighted PageRank algorithm used in the three tasks. Algorithm 1 *weighted-PageRank(items)* takes the nodes (*items*) and returns their ranking.

Data Description

Publication Ranking We are using the Association of Computational Linguistics (ACL) corpus[3]. It contains 21k papers, 110k internal citations.

Extractive Document Summarization To evaluate the summarization technique we used the Document Understanding Conferences (DUC) 2002 dataset[4]. The version we used has 104 articles, each around 400 words. Reference summaries are given, each 200 words. So the task will be to summarize a 400-word article into a 200-word one.

[3]http://clair.eecs.umich.edu/aan/index.php

[4]http://www-nlpir.nist.gov/projects/duc/data/2002_data.html

Algorithm 1 weightedPageRank(items)

```
DG ← new directed graph
for i in items do
  Add node i to DG
  s ← new empty list
  for j in Out(i) do
    s[j] ← similarity score between (i, j)
  end for
  if sum(s) > 1.0 then
    normalize(s)
  end if
  for j in Out(i) do
    add node j to DG
    add edge between (i, j) with score s[j] to DG
  end for
end for
PR ← perform a weighted pagerank to DG
return PR
```

Keyword Extraction In order to evaluate our technique we use the Inspec dataset[5] (Hulth 2003). It contains 500 scientific paragraphs and the keywords for each paragraph are extracted manually by annotators.

Results

Publication Ranking

Evaluating this task is hard because it is highly subjective. Claiming a paper is influential or not is very subjective, and manual annotation for this task shows differences even for experts. One idea is to assume all the publications that won *best paper award* to be influential. For the ACL dataset we have 46 best papers. We test using LDA, document vectors for edge weighting versus the default unweighted edges. We calculated the recall, precision, F1-score and NDCG for each technique. An ideal ranking (oracle) will place the 46 best papers in the top 46 places. Let the set of best papers be BP, let the ranked publications sorted by any technique be PR, and let the top 46 rank of PR be $RP_{|BP|}$.

$$tp = |p \in BP, p \in PR_{|BP|}|$$
$$fp = |p \notin BP, p \in PR_{|BP|}|$$
$$fn = |p \in BP, p \notin PR_{|BP|}|$$
$$tn = |p \notin BP, p \notin PR_{|BP|}|$$

Where tp, fp, fn, tn are the true positives, false positives, false negatives and true negatives respectively. Then we can calculate the recall, precision and F1-score.

The NDCG of p_i where p_i is one of the best papers

$$NDCG(p_i) = \begin{cases} 1 & p_i \text{ ranked at top of } |BP| \\ \frac{\frac{1}{log(r)}}{\sum_{k=|BP|+1}^{r} \frac{1}{log(k)}} & \text{if } p_i \text{ is ranked at } r \end{cases}$$

Now the total $NDCG = \frac{\sum_{p_i \in BP} NDCG(p_i)}{|BP|}$

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

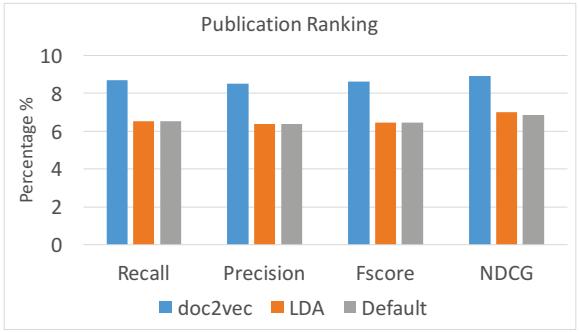

Figure 2: Publication Ranking Evaluation

Figure 2 shows that our weighted PageRank algorithm using document vectors outperforms default unweighted PageRank with 2%. It also shows that using document vectors as similarity measure is superior to using LDA.

Extractive Document Summarization

Using the DUC dataset we compare two similarity techniques discussed, the Min/Max/Sum sentence representation and the word overlap. We're using ROUGE (Lin 2004) as the metric for measuring the quality of the translations

- ROUGE-N: N-gram co-occurrence statistics. We used N=1,2,3,4.
- ROUGE-L: Longest Common Subsequence (LCS). It identifies longest co-occurrence in a sequence of n-grams
- ROUGE-W: Weighted LCS-based statistics that favor consecutive LCSes .
- ROUGE-S: Skip-bigram co-occurrence statistics. Skip-bigram is any pair of words in their sentence order.
- ROUGE-SU: Skip-bigram plus unigram-based co-occurrence statistics.

Figure 3 shows the ROUGE metric against the window size for using different similarity metrics: vector representation, word overlap and unweighted PageRank. Surprisingly, using word overlap as a similarity measure outperformed using word vectors.

Keyword Extraction

In order to evaluate our technique, we use the Inspec dataset (Hulth 2003). It contains 500 scientific paragraphs and the keywords for each paragraph are extracted manually by annotators to be used as ground truth. (Mihalcea and Tarau 2004) used this dataset in keyword extraction by creating an unweighted undirected graph for the input text and selecting the top k words with the highest PageRank scores. Below are the hyper-parameters used in the experiments.

- WINDOW: Basically words that are less than "window" apart are considered neighbors; i.e. this parameter determines which words in the text between which to draw edges. We use a window around each word so that words in range [p-w:p+w] will have an edge with the current word, where p is the position of the current word and w is the window size
- K: The ratio between the number of top rated words to consider as extracted keywords and the total number of input words.
- POS: To filter out words by part of speech tags. In our experiments (following the previous work) we only consider NN (Noun, singular), NNS (Noun, plural), JJ (adjectives).
- SIM: The similarity measure used between edges.

For the sake of comparison, We used same configuration as (Mihalcea and Tarau 2004) [WINDOW=2, K=0.3, POS=NN/NNS/JJ] with only adding weights to the edges of the graph using the similarity measures discussed previously. The results are reported in Figure 4, which shows the vector based approach with cosine similarity is superior to other similarity measure and shows 0.5% increase in F1-score over the unweighted approach. Now, we will change the hyper-parameters used in the experiments to see their effect on the performance of different techniques. Figure 5 shows the effect of changing the window on the unweighted graph and the vector based weighted graph with cosine similarity, which reveals that word-to-word similarity weighted graphs based on word embeddings is superior to the unweighted approach with around 1.5% gain in F1-score.

Discussion

In this paper, we discussed a hybrid graph- and NLP-based techniques to summarize influential publications. We proposed a framework to rank influential papers, then to summarize those papers and finally to extract important keywords. We represented items to be ranked as nodes, and relationships as edges weighted with similarity measure between connected items. We used textual features to give different weights to citations based on the contextual similarity between the two papers. We experimented with two textual similarity techniques: the first is to embed papers in a multidimensional space, then calculate the cosine similarity between them; the second is to analyze the topics of each paper with LDA and the similarity will be how much the topics of the two papers overlap. We showed that by using textual features with citation graph information we were able to give higher ranks for important papers (those which won best paper award) than using citation information solely with no less than 2%. Using the same framework we boosted the performance of keyword extraction using a graph-based approach with no less than 1.5%.

However, For the extractive document summarization task, using words overlap as similarity measure outperformed sentence vector similarity evaluated against the sample of DUC-2002 dataset. Possible extensions to the work here is to investigate edge-wise personalized PageRank as a solution to the convergence problem discussed in the methodology section. Also, finding other methods to test the influential paper ranking will provide a better test-bed for bench-marking other techniques.

References

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

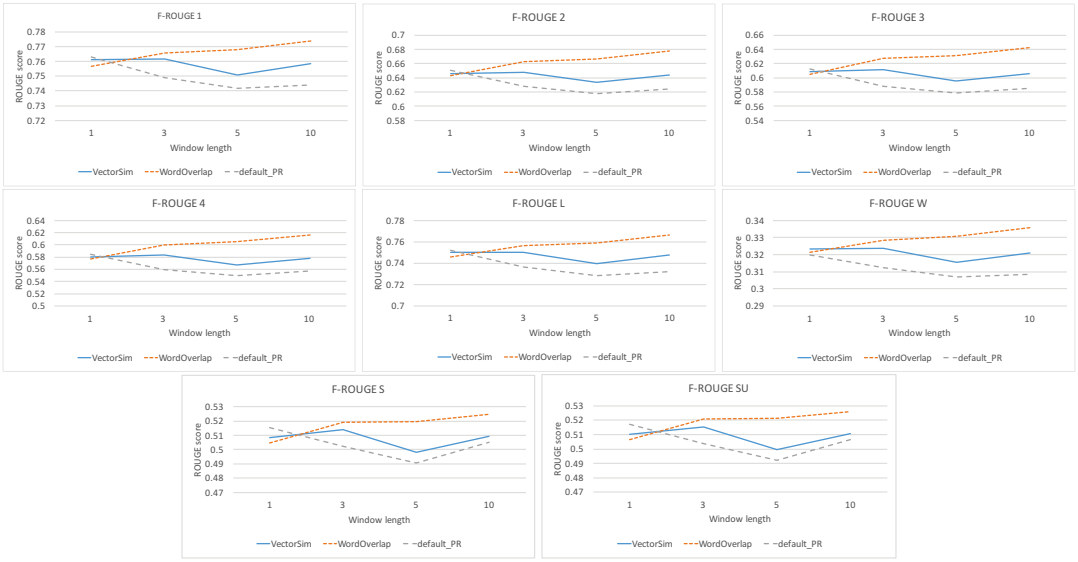

Figure 3: ROUGE scores using vector based similarity and word overlap similarity with changing the neighborhood window.

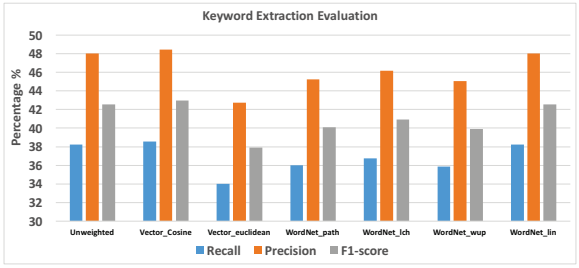

Figure 4: Results on Inspec data set using different similarity measures

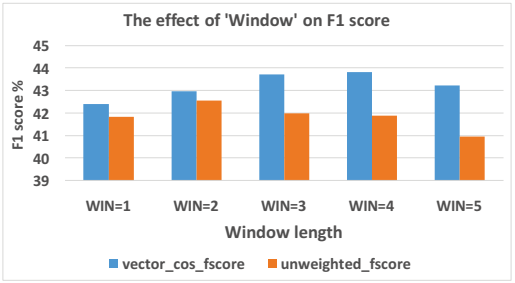

Figure 5: Effect on changing WINDOW on the F1-score of unweighted graph and the weighted graph using cosine similarity between word vectors

Gerrish, S., and Blei, D. M. 2010. A language-based approach to measuring scholarly impact. In *ICML*, volume 10, 375–382.

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

Le, Q. V., and Mikolov, T. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Leacock, C., and Chodorow, M. 1998. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database* 49(2):265–283.

Lin, D. 1998. An information-theoretic definition of similarity. In *ICML*, volume 98, 296–304.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8.

Mihalcea, R., and Tarau, P. 2004. Textrank: Bringing order into texts. Association for Computational Linguistics.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Ramage, D.; Manning, C. D.; and McFarland, D. A. 2010. Which universities lead and lag? toward university rankings based on scholarly output. In *Proc. of NIPS Workshop on Computational Social Science and the Wisdom of the Crowds*. Citeseer.

Sun, Y., and Giles, C. L. 2007. Popularity weighted ranking for academic digital libraries. In *European Conference on Information Retrieval*, 605–612. Springer.

Wu, Z., and Palmer, M. 1994. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, 133–138. Association for Computational Linguistics.