Competing Algorithm Detection from Research Papers

Soumyajit Ganguly
Center for Data Engineering
International Institute of Information Technology
Hyderabad, India
soumyajit.ganguly@research.iiit.ac.in

Vikram Pudi
Center for Data Engineering
International Institute of Information Technology
Hyderabad, India
vikram@iiit.ac.in

ABSTRACT

We propose an unsupervised approach to extract all competing algorithms present in a given scholarly article. The algorithm names are treated as named entities and natural language processing techniques are used to extract them. All extracted entity names are linked with their respective original papers in the reference section by our novel *entity-citation* linking algorithm. Then these *entity-citation* pairs are ranked based on the number of comparison related cue-words present in the entity-citation context. We manually annotated a small subset of DBLP Computer Science conference papers and report both qualitative and quantitative results of our algorithm on it.

1. INTRODUCTION

It is often the case in the domain of computer science that a new scholarly article introduces some novel algorithm for solving a particular existing problem. In the process of doing so, the authors show that their new algorithm outperforms the current state of the art. Browsing online repositories like Google Scholar or CiteSeer are common ways for searching related papers for a particular problem. While browsing, a researcher would have to manually visit the references section and try to find all other competing papers from the long list. Reviewing only the experiments section might help identity the algorithms but it still remains quite a time consuming task for repeating this process everytime. The motivation of our work is to fully automate this entire process and provide an instant list of competing algorithms to the current problem and their corresponding original papers from the reference section. We propose a linguistic approach to solve this problem and our framework is based on first analyzing the entire content of the paper as input and then producing a ranked list of algorithm names and reference papers as output. Using our framework, we can list all the competing algorithms of a given paper along with their references in a sorted order which would help a fellow researcher who is browsing the given paper.

CODS '16, March 13 - 16, 2016, Pune, India

ISBN 978-1-4503-4217-9/16/03…$15.00

DOI: http://dx.doi.org/10.1145/2888451.2888473

Early works on citation classification in scholarly articles were investigated by Nanba and Okumura [3] who came up with a citation classification scheme having three categories - *contrast*, *compatability* and *background*. They introduced cue words specific to each category which we use later in our algorithm for comparison score. Considerable work has been done to aid fellow researchers [1] when they are browsing any particular subdomain in computer science but most of these are graph based approaches which only use information like authors, citations and title. Graphs fails to capture the motivation behind citations. Teufel and Dong et. al. [4] used machine learning techniques to classify citations by sentiments. We expect that the entire content of a paper carries rich information which is not captured in citation graph networks. To the best of our knowledge, no work has been published directly on mining competing algortihms by exploring the full textual content of a scholarly article.

2. PROPOSED APPROACH

We take a single scholarly article as input which will have a list of cited papers in its reference section. Here we assume that the authors might be introducing a new algorithm in this article and would compare it with existing techniques that would be mentioned in the list of cited papers. We give each reference paper a score on the basis of this comparison. Finally we rank all of them in decreasing order of their comparative score. *Our intuition is that algorithm names when written in a scientific paper by any author, it is syntactically and semantically similar to traditional named entities.* Thus we hope to capture these algorithm names by using named entity extraction procedures. Our proposed model is described stepwise in subsequent subsections.

Named entity extraction and filtering: After accquiring raw text, we segment it into individual sentences and apply named entity recognition technique to gather all unique named entities in the paper. We get quite a number of false positives at this stage since not all named entities are algorithm names and the recognition techniques are trained on plain English literature corpus. A look-up table is constructed to remove all the corporation and institution names. All standard datasets are removed next by looking at the keyword "dataset" directly after the mention of entity. We also prune out entities outside the range 2 and 30 characters. We ignore the removal of author names in this step and rely on our entity-citation linking procedure shown next by which we attempt to prune all unnecessary named entities.

Entity Citation linking: All citations in the paper are recognized using resular expressions. Only the sentences

which contain both *named entities* and *citations* are considered from here on. The entity-citation matching function iterates over all possible combinations of entities and citations and get their *closeness score* which is the string distance in characters between the named entity occurrence and its citation. We check for the presence of other algorithm names inside this region and eliminate those. After the scoring step, we perform a two-step pruning. It first takes all the citations and keeps a list of the closest entity per citation. Then it takes the remaining entities and keep only the closest citations per entity. By this linking process, we hope to eliminate all noisy enitites and other citations which are not algorithm name related. As an output of this step, we get a list of tuples where each element contains an unique entity matched with its citation.

Comparative cue scoring: In this step we have a complete list of algorithm mentions along with their respective cited papers. We gather a set of comparison related cue-words for ranking a competing algorithm in a paper. The motivation for using cue words are based on [2] where the authors tried to extract comparative sentiments in text data. We then score each named entity by the number of cue words present in its context for all occurrences in the paper. After cue-word scoring per entity, we rank them in descending order of their scores.

3. EXPERIMENTS AND RESULTS

Our experiments are conducted on a small collection of papers from DBLP dataset which cover varying fields like *natural language processing, computer vision, databases, networking*. We use regular expressions to extract citation occurences from papers that are written in the Springer LNCS / LNAI format. A sample of the regular expression used by us in Python is shown below. This example is for capturing the 10^{th} reference. This is iteratively applied for all the references and it can also extract citations where multiple occur together. `(\[[^\]]*)\b(10)\b(?=[^\]]*\])`
Regular expressions are also used for named entity extraction where authors use short-forms and punctuation marks. An example would be "*Support Vector Machines **(SVM)***."

A total of 31 papers had been manually annotated as the ground truth data for testing our algorithm. We based the decisions only on the input paper and the respective author's view of competing algorithms in the input paper. For conversion of paper format from *.pdf* to *.txt* we used the free and popular tool Apache PdfBox[1]. We also use the NLTK[2] package for extracting all named entities in the paper.

The result for a representative paper from our annotated dataset can be seen in table 1. The first column in the table contain serial numbers from the reference section which the respective named entity from second column link to. The third column gives the total cue score for the named entity and citation combined. The last column is the ground truth data which is compared with the first column and states whether this reference paper is a competing algortihm or not. The first FALSE result *'UAC'* is observed to be the own algorithm of the authors in that paper which our current approach is unable to filter out. Following 4 results are all correct. Towards the end of the list, as the cue-score decreases we find more false-positives. In our entire dataset,

[1]http://euske.github.io/pdfminer/index.html
[2]http://www.nltk.org/

Table 1: Algorithm results on paper 'An associative classifier for uncertain datasets, 2012.'

An associative classifier for uncertain dataset			
ref. no.	named entity	cue-score	output
21	UAC	53	FALSE
13	UCBA	20	TRUE
12	uHARMONY	14	TRUE
11	uRule	9	TRUE
8	DTU	7	TRUE
14	CBA	5	FALSE

we got an initial recall of **0.75** which means that out of all competing algorithms we could retrieve **75%** of them and among the **top 3** results we achieved a precision of **42%**. The cause for a low precision value is that out of the total 31 papers we chose to annotate, only 24 are competitive. The authors of the other 7 papers are not comparing their results with any other pre-existing algorithm for the same problem, instead they introduce a novel problem and propose a first solution for it. In these kinds of papers our citation entity linking algorithm give sub-optimal results. Another reason for low precision is that pdf to text conversion and named entity recognition are not perfect and their individual errors add up in our full process.

We had to follow an unsupervised approach throughout our work due to the lack of quality and quantity of annotated training data. Supervised machine learning algorithms would help increase our accuracy for classifying citation sentences. We can use another scholarly article text classifier in the pre-processing stage and filter the non-competitive papers. Not being confined to the given paper for textual information and exploring all the papers mentioned in the reference section can give us better quality named entities.

In this paper we demonstrated a procedure for finding competing algorithms for scholarly articles. Most of the available research works in this area do not consider the texttual content. We show that there are a lot of information available when we process the entire text of a scholarly article, competing algorithms just being a start.

4. ACKNOWLEDGMENTS

This work was supported in part by research grants from the Department of Biotechnology, Government of India.

5. REFERENCES

[1] N. Agarwal, K. Gvr, R. S. Reddy, and C. P. Rosé. Scisumm: a multi-document summarization system for scientific articles. Association for Computational Linguistics, 2011.

[2] C. Dong and U. SchÃd'fer. Ensemble-style self-training on citation classification. In *IJCNLP*. The Association for Computer Linguistics, 2011.

[3] H. Nanba and M. Okumura. Towards multi-paper summarization using reference information. 1999.

[4] S. Teufel, A. Siddharthan, and D. Tidhar. An annotation scheme for citation function. In *SIGdial*. Association for Computational Linguistics, 2006.