SmartCiteCon: Implicit Citation Context Extraction from Academic Literature Using Supervised Learning

Chenrui Guo
Haoran Cui
Wuhan University
Wuhan, Hubei, China
chenruiguo@whu.edu.cn
haoran.cui@whu.edu.cn

Li Zhang
Jiamin Wang
Wei Lu
Wuhan University
Wuhan, Hubei, China
weilu@whu.edu.cn

Jian Wu
Old Dominion University
Norfolk, VA, USA
jwu@cs.odu.edu

Abstract

We introduce SmartCiteCon (SCC), a Java API for extracting both explicit and implicit citation context from academic literature in English. The tool is built on a Support Vector Machine (SVM) model trained on a set of 7,058 manually annotated citation context sentences, curated from 34,000 papers in the ACL Anthology. The model with 19 features achieves F_1=85.6%. SCC supports PDF, XML, and JSON files out-of-box, provided that they are conformed to certain schemas. The API supports single document processing and batch processing in parallel. It takes about 12–45 seconds on average depending on the format to process a document on a dedicated server with 6 multithreaded cores. Using SCC, we extracted 11.8 million citation context sentences from ∼33.3k PMC papers in the CORD-19 dataset, released on June 13, 2020. The source code is released at https://gitee.com/irlab/SmartCiteCon.

1 Introduction

Citations are ubiquitous in scientific publications. With proper citations, statements in research papers are supported by existing works, and readers obtain relevant information beyond the current paper. Citations also form graphs, which provide unique models for ranking, sentimental classification, and plagiarism detection. Therefore, citation analysis plays an important role in helping to understand the deep connection between literature. Accurate citation context recognition is the prerequisite of many downstream applications. Recently, citation context, the text segment that appears around the citation mark in the body text, has been used for enhancing and improving keyphrase extraction (Caragea et al., 2014) and document summarization (Cohan and Goharian, 2015).

There are two types of citation context. Explicit citation contexts (ECC) are sentences containing citation marks. Each citation thus corresponds to one explicit citation context sentence. Implicit citation contexts (ICC) are sentences that are semantically relevant to the cited articles but do not contain citation marks. ICC may appear before or after but may not immediately precede or follow the ECC sentence. One paper could be cited multiple times and each time may have different citation contexts. In the example below, the ECC, containing the citation mark "(Ma et al. 2004)", is highlighted in green. The ICC sentences are highlighted in yellow. The nonhighlighted sentence is not a citation context for the given citation.

> We investigate the impact of semantic constraints on statistical word alignment models as prior knowledge. In (Ma et al. 2004), bilingual semantic maps are constructed to guide word alignment. The framework we proposed seamlessly integrates derived semantic similarities into a statistical word alignment model. And we extended monolingual latent semantic analysis in bilingual application.

Most existing tools extract ECC, i.e., sentences containing citation marks. Although the results are highly relevant, the method omits ICC if the author uses multiple sentences to summarize the results. To our best knowledge, there are no off-the-shelf tools dedicated to ICC extraction. Unlike ECC sentences with citation marks, the lack of explicit marks makes citation context recognition challenging.

In this work, we develop a Java API that implements a supervised machine learning model trained on 7058 manually labeled sentences to extract both ECC and ICC. The model achieves an F_1-measure of 85.6%. The Java API can be deployed on a local machine or as a web service.

2 Related Work

Several citation context extraction methods have been developed. In Nanba and Okumura (1999), the scope of the citation context covered several consecutive sentences before and after the sentences with citation marks (i.e., citation sentence), identified based on a referential relationship with the citation sentence. In another work, Markov model was used for identifying citation context (Qazvinian and Radev, 2010). Sugiyama (2010) described a support vector machine (SVM) and maximum entropy (ME) model for identifying citation sentences using shallow features such as proper nouns and contextual classification of the previous and next sentence (Sugiyama et al., 2010). They found that the performances of SVM and ME do not exhibit significant differences. The positive samples were selected as sentences including citation marks using regular expression matching, ICC extraction was not covered.

ParsCit is an open-source software commonly used for citation parsing and citation context extraction (Councill et al., 2008). ParsCit parses citation strings using a Conditional Random Field (CRF) model. The citation context extraction was performed by extracting a fixed window size of 200 characters on either side of the citation mark. GROBID (Lopez, 2009) is a library to extract information from scholarly documents. The documentation reports the F_1-measure of citation context resolution is around 75%, which counts both the correct identification of citation marks and its correct association with bibliographic references.

In summary, existing citation context extraction tools focus on ECC but ignore ICC, the latter of which includes more sentences semantically related to the cited papers.

3 Supervised Machine Learning Model

Our system is based on a supervised machine learning model proposed in Lei et al. (2016), which classifies a sentence into ICC and non-ICC.

We adopted the ground truth built by Lei et al. (2016) containing 130 articles from 34,000 computational linguistics conference proceedings in ACL Anthology. The original PDF files were converted to XML format using OCR (Schäfer and Weitz, 2012). The training set was labeled by 13 graduate students majoring in information management. The labeling agreement was tested using Cohen's Kappa Coefficient ($\kappa = 0.937$). The final ground truth contains 3,578 positive and 3,480 negative samples. The preprocessing uses Apache OpenNLP for sentence segmentation. Citation marks are identified using regular expressions. Citation marks are then removed, and the original sentences are converted into regular sentences for following analyses such as part-of-speech (POS) tagging. Each sentence is represented by up to 19 features of four types (Table 1). The best model using all features achieves 86% F_1-measure in a the 10-fold cross validation. The SVM outperformed CRF by about 5% in F_1-measure (Table 2).

4 Architecture

The SCC system completes the extraction in four steps (Figure 1): (1) file type recognition, (2) preprocessing, (3) feature extraction, and (4) sentence classification. The output is a JSON file containing ECC and ICC and other citation-related information. The API was written based on the Springboot framework in Java. The machine learning model was implemented with WEKA.

4.1 File Type Recognition

SCC first recognizes the uploaded file type. For a PDF file, SCC invokes GROBID and converts it to an XML file under the TEI schema. If an XML file is uploaded as input, SCC checks whether the schema is in compliance with TEI or PloS ONE schema and passes it to corresponding preprocessors. If a JSON file is uploaded, it checks if it is in compliance with the S2ORC schema, published by Semantic Scholar (Lo et al., 2020). We apply Apache Tika to identify file format. Other format of data files will not be processed.

4.2 Preprocessing

The preprocessing step reads files passed from the last step with customized preprocessors depending on the schema and prepares a canonicalized XML for feature extraction. This step includes the following modules.

4.2.1 Tag removal

This module involves removing irrelevant tags from the DOM structure in the XML file. For example, in the PloS ONE XML files, the `<fig>`, `<sub>`, and `<italic>` tags used for marking up figures, superscripts, and italic font are all moved. Only the text inside these tags are retained. The `<xref>` tags mark positions of citations, which will be used for

#	Features	Categories
1	Distance to the citation sentence	Location
2	In the same paragraph as the citation sentence	Location
3	Include any citation marks	Location
4	The preceding sentence is not a citation sentence	Location
5	The following sentence is not a citation sentence	Location
6	The preceding sentence is the first in paragraph	Location
7	Is the first sentence in the paragraph	Location
8	Section the sentence is in	Location
9	Is the last sentence in the paragraph	Location
10	Trigram Jaccard similarity	Content
11	Bigram Jaccard similarity	Content
12	Unigram Jaccard similarity	Content
13	Include author names	Reference
14	Include any words in the citation sentence	Reference
15	Include He/She/It or their variants	Reference
16	Include Lexical hooks (Murray, 2015)	Reference
17	Include Work Nouns (Murray, 2015)	Reference
18	Number of citation marks	Type
19	Include certain conjunction	Structure

Table 1: Features of the SVM model. A citation sentence is the sentence containing a citation mark.

Model	Precision	Recall	F1-measure
SVM_19	85.6%	85.6%	85.6%
CRF_19	82.2%	79.9%	80.8%

Table 2: Evaluation of SVM and CRF models on 19 features.

restoring citations. We use a separate data structure to store the positions of `<xref>` tags before removing them.

4.2.2 Sentence segmentation

We compared five commonly used sentence segmentation tools, including the Pragmatic Segmenter by Kevin Dias[1], lingpine[2], NLTK[3], a regular expression parser, and the Stanford CoreNLP (Manning et al., 2014) sentence splitter. The golden standard contains 52 sentences provided by Kevin Dias, which covers most possible sentence forms. According to Dias' comparison, the Pragmatic Segmenter receives an accuracy of 98% and the Stanford CoreNLP's accuracy is 59.6%. In our experiments, the accuracies for Lingpipe, NLTK, and regular expression parsers are 61.5%, 50.0%, and 38.5%, respectively. The Pragmatic Segmenter is implemented by Ruby on Rails. To make our API less dependent on a second programming language, we decided to employ Lingpipe for sentence segmentation. We select up to five sentences before and after the current citation sentence as the candidates for classification. This covers almost all sentences that could be classified as ICC.

4.2.3 Canonicalization

Because the input XML may have different schemas, this module takes the processed documents from the above modules and transforms them into a unified schema for feature extraction. The canonicalized schema defines new IDs for chapters, paragraphs, sentences, and citations. The canonicalized XML also includes whether the current sentence contains citation marks.

4.3 Feature Extraction and Text Classification

This step extracts 19 features (Table 1) from the canonicalized XML files and represents each candidate sentence as a vector saved in Livsvm files[4].

[1] https://github.com/diasks2/pragmatic_segmenter

[2] http://www.alias-i.com/lingpipe/

[3] https://www.nltk.org/

[4] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

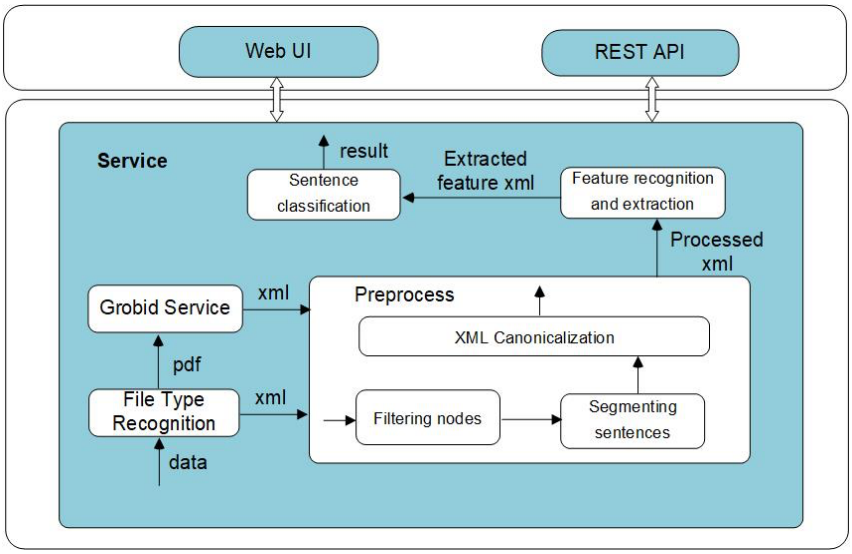

Figure 1: SmartCiteCon architecture.

The SVM model classifies each sentence and outputs a binary indicating whether a sentence is ICC or not. The output JSON file contains citation marks and their positions, citation sentences, and sentences classified as ICC.

4.3.1 User Interfaces

Users can install SCC on a local machine. The API interface supports 3 modes:

1. Single document mode – using the `/extract` service;
2. Batch extraction model with files zipped and transferred through TCP/IP – using `/batchExtract`;
3. Local extraction model with files retrieved from a local directory – using `/localExtract`.

In the single document and batch extraction modes, the API will return JSON objects and execution status. For the local extraction mode, the API will return the execution status and the results will be saved in JSON files.

5 SCC API Performance

We test the SCC API on a computer with 16GB RAM and an Intel Core i7-8570H CPU@2.20GHz, which has 6 hyperthreaded cores (12 threads in total). In a preliminary experiment, we compare the runtime of processing 10 XML documents using a single process under different JVM heap sizes. The runtimes corresponding to 12GB, 8GB, 4GB, and 2GB are 23.8 min, 12.7 min, 7.3 min, and 7.6 min, respectively. Higher heap does not boost processing speed probably due to garbage collection. Based on the results, in the following experiments, 4GB heap was allocated to JVM. The experiments were set to extract citation context from randomly selected documents in different formats. The datasets include 10 PDF documents from PLoS ONE, 10 XML documents corresponding to the PDF documents, and 10 JSON documents from the CORD-19 dataset. We monitor the system using Jprofiler (version 11) and calculate the median time it takes for processing one document as we vary the number of processes N_p. Figure 2 shows that the CPU utilization increases from about 10% and saturates when N_p reaches 8. The memory utilization climbs up slowly as N_p increases but are mostly well below the maximum allocated heap, because processed documents are not stored in memory anymore. The average processing time for all three types gradually decreases as N_p increases but in general, it takes longer to process PDF files than JSON and XML files. The maximum and minimum processing time are shown in Table 3. The runtime can be further reduced by running the API on a computer with more processes on a multicore server. On average, JSON files take the least time to process.

6 Extracting Citation Context from CORD-19

SCC is different from similar tools such as ParsCit and GROBID in that it extracts both ECC and ICC.

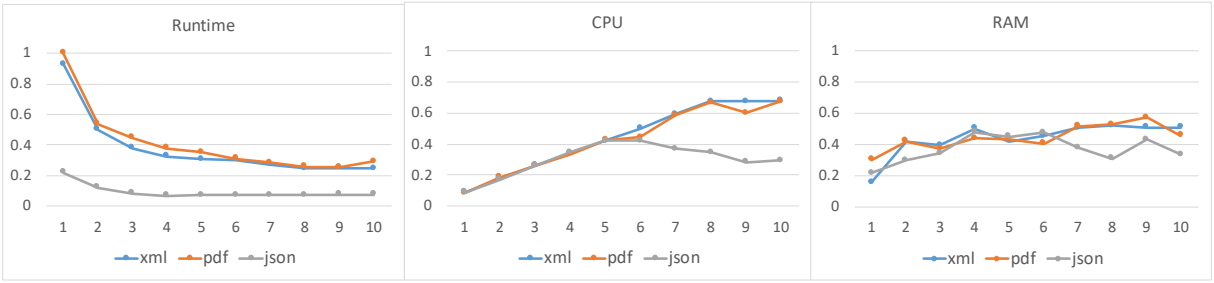

Figure 2: The performance of SCC on a multicore computer. Runtime is normalized at the 177 seconds; the middle panel shows the CPU utilization monitored by Jprofiler; the right panel shows the memory utilization normalized at 4GB.

XML		PDF		JSON	
Max	Min	Max	Min	Max	Min
164	43	177	45	39	12

Table 3: Runtime in seconds for different document formats. The maximum and the minimum runtime are achieved at $N_p = 1$ and $N_p = 8$, respectively.

We apply SCC and extract ECC and ICC from the **CORD-19** dataset. CORD-19 is an open-access dataset compiled by Allen Institute of Artificial Intelligence about COVID-19, SARS, MERS, and related keyphrases conforming to the S2ORC schema (Lo et al., 2020). We downloaded the data released on June 13, 2020 including 50,818 and 69,646 full text papers under the PMC and the PDF folders respectively. The PMC folder contains full-text files obtained by parsing JATS[5] XML files available for PMC papers using a custom parser, generated to the same target output JSON format. This resulted in 1,605,695 ECC and 10,215,848 ICC sentences from 33,319 documents. A fraction of documents was not processed due to the lack of citation marks and runtime exceptions.

SCC code is released at `https://gitee.com/irlab/SmartCiteCon`. The dataset is available on Microsoft OneDrive with a link on the code repository.

7 Lessons Learned

The results in Table 3 indicate that SCC takes about 45 seconds on average to process a PDF document, which is still relatively slow. Using Jprofiler, we found that more than 90% time was spent on preprocessing, specifically canonicalization, followed by sentence classification (for XML and JSON) or file type recognition (for PDF). The bottleneck is partially attributed to the word tokenization and POS tagging in the Stanford CoreNLP API. One way to mitigate this problem is to use the Stanford CoreNLP Server[6]. Alternatively, we can use Stanza (Qi et al., 2020), the successor of Stanford CoreNLP. Empirical results have shown that it is faster than CoreNLP in several NLP tasks. Stanza was written in Python, but we can develop a RESTful service. The slowness can also be attributed to the poor garbage collection in Java, which can impact CPU usage massively. A more systematic and fine-grained profiling is needed to diagnose the root cause of this problem.

8 Conclusions and Future Works

We developed SmartCiteCon (SCC), a Java API to extract explicit and implicit citation context from academic literature. The API implements an SVM model achieving an $F_1 = 85.6\%$. SCC accepts XML (in PLoS ONE schema or GROBID schema), PDF, and JSON (in S2ORC schema) formats. The output of SCC is a JSON file containing marked citation contexts and paper metadata if available. We applied SCC on the PMC subset of the CORD-19 dataset and obtained about 11.8 million citation context sentences in which 10.2 million are implicit citation context.

One limitation of SCC is that the model was trained on papers in computational linguistics, so more careful evaluation and feature distribution analysis should be performed when applying the model to other domains. In the future, we will explore word embedding models to enrich semantic features and improve scalability by overcoming performance bottlenecks.

[5] `https://jats.nlm.nih.gov/`

[6] `https://stanfordnlp.github.io/CoreNLP/corenlp-server.html`

Acknowledgments

We thank Zikun Feng for setting up a web-based user interface and Shengwei Lei for constructive discussion.

References

Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1435–1446.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 390–400, Lisbon, Portugal. Association for Computational Linguistics.

Isaac Councill, C Lee Giles, and Min-Yen Kan. 2008. ParsCit: an open-source CRF reference string parsing package. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

Shengwei Lei, Haihua Chen, Yong Huang, and Wei Lu. 2016. Research on automatic recognition of academic citation context. *Library and Information Service*, 60(17).

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

Patrice Lopez. 2009. GROBID: combining automatic bibliographic data recognition and term extraction for scholarship publications. In *Proceedings of the 13th European Conference on Research and Advanced Technology for Digital Libraries*, ECDL'09, pages 473–474, Berlin, Heidelberg. Springer-Verlag.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Jonathan Murray. 2015. Finding implicit citations in scientific publications. Master's thesis, KTH Royal Insitute of Technology.

Hidetsugu Nanba and Manabu Okumura. 1999. Towards multi-paper summarization using reference information. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, page 926–931, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Vahed Qazvinian and Dragomir R. Radev. 2010. Identifying non-explicit citing sentences for citation-based summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 555–564, Uppsala, Sweden. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, ACL 2020, Online, July 5-10, 2020*, pages 101–108. Association for Computational Linguistics.

Ulrich Schäfer and Benjamin Weitz. 2012. Combining OCR outputs for logical document structure markup. technical background to the ACL 2012 contributed task. In *Proceedings of the Special Workshop on Rediscovering 50 Years of Discoveries@ACL 2012, Jeju Island, Korea, July 10, 2012*, pages 104–109. Association for Computational Linguistics.

K. Sugiyama, T. Kumar, M. Kan, and R. C. Tripathi. 2010. Identifying citing sentences in research papers using supervised learning. In *2010 International Conference on Information Retrieval Knowledge Management (CAMP)*, pages 67–72.