PaperMiner: An Assistant to Extract Information from Research Papers for Knowledge and Career Development

Zan Huang*
Georgia Institute of Technology
huangzan@gatech.edu

Kaiwen Luo*
Georgia Institute of Technology
kluo37@gatech.edu

Kai Li*
Georgia Institute of Technology
kaili@gatech.edu

Jiaji Liu*
Georgia Institute of Technology
jliu808@gatech.edu

Maiqi Ding*
Georgia Institute of Technology
mding41@gatech.edu

ABSTRACT

We built a research paper investigation platform for computer science major students to embrace tons of papers published every year, especially those related to artificially intelligence, which are hard to search and read in the pure manual approach as the corpus is growing too fast. For helping students searching papers, know venues and researchers, get suggestions for the paper they might be working on. We went through all phases including data collection, data mining and data visualization, present a multi-function platform, to help those who are willing to read papers for knowledge and career development. Compared with existing platforms like google scholar, we tried to go deeper into computer science sub-fields instead of go wider for all subjects and explore the whole research paper corpus. We also focus on text mining instead of citation network analysis as there are more and more open-access papers and better tools for nature language processing now. Finally, our platform is lightweight which be open-sourced to help more people to **read** papers by **coding** for productivity.

*Authors contributed equally to this work.

1 INTRODUCTION AND PROBLEM DEFINE

Academic papers are great resources for knowledge discovery[9], data mining[7] and network analysis[10]. Also, they are valuable in guiding personal knowledge and career development[18].

Nowadays, the corpus of academic papers is huge with hundreds of millions of publications by authors from tens of thousands of institutions published at conferences and journals on a wide range of topics[6].

At the same time, the number of papers published each year keeps increasing[2]. On the other hand, academic papers are playing a more and more important role in student's early careers for building up a personal knowledge base.

However, the huge number of publications makes it hard for individuals to find relevant literature on a topic. Tools (Arxiv, sotawhat)[1, 4] have been developed to provide search services to reduce the pain by filtering out related papers. Other tools (SOTA, CSRanking)[3, 5] put more focus on providing categorical rankings based on the corpus.

With the advance in nature language processing[8] and more open access papers published in the computer science field, we have richer corpus for text mining compared to citation network analysis for investigating research papers. Existing tools like

Google scholar[16] and Aminer[15] put more focus on whole academic paper corpus-based citation network analysis. Instead, we focus on a computer science field, put more focus on text mining for information extraction, go deeper on a subset of the corpus for new findings.

Problem Define. In this work, we present PaperMiner, a novel computer science literature search platform for college students. PaperMiner combines literature search, academic data presentation, interactive visualization, and open-source API for customized research. PaperMiner contains three part: data

2 SURVEY

A lot of great works [11, 14–16] have been done based on the corpus to study the citation network, topics, and trends of research. Including applying NLP and reinforcement learning to improve the experience for exploring academic paper[17].

Google Scholar and Microsoft Academic are the most widely used literature search platform. They are powerful for searching papers but lack the summary of the data, i.e. researcher collaboration, current trends, etc. AMiner is another platform which combines literature search with data summarization and visualization. It is a data-driven platform that presents analysis results along with research paper. For insights of research advancement and collaboration between experts, to support users retrieving academic papers multidimensionally.

AMiner is a powerful academic search platform which satisfies most people's needs, but users are still passive in literature survey which may not meet personalized needs. The searching algorithms used so far rest mostly on searching for titles, authors etc. rather than topics. Furthermore, for college students, it may be better to show recent research trends and recommend representative words besides providing a search bar, best if they can DIY in data analysis based on open-source projects.

We propose PaperMiner, a light-weight and reusable project, designed for students interested in computer science and want to explore the academic papers for their own research interests (through CS topic trends, top influencers and institutes on subfeilds, etc.). We collect data from multiple sources like dblp[11] and arxiv[12], perform data analysis and text mining on pre-processed data and finally integrate and visualize the results by a web app.

3 PROPOSED METHOD

3.1 Intuition

Comparing with other literature search platforms like Google Scholar, Microsoft Academic and AMiner. PaperMiner combines the literature search with academic data presentation. Apart from presenting the historical data, like the number of publication and citation, PaperMiner also present insights like trends and topics extracted by machine learning algorithms. PaperMiner focuses on the student community and computer science area, providing more tools for helping students' future knowledge and career development.

3.2 Data Collection

We collect data from different sources and perform data integration for later usage. Namely, we collected dblp[11] computer science publication dataset for tracking more than 5,000,000 publications, the dataset could be downloaded as one big XML file from official website[1]. Proceedings of some important conferences are well organized and publically available online, i.e. NeurIPS[2]. We scraped about 20,000 papers of 11 computer science conferences from 4 different sources(nips, aclweb, thecvf, jmlr)till 2020 including (title, venue, year, authors, abstract) information along with corresponding PDF file which got turned into a TXT file for potential use. Arxiv and Google Scholar are used for searching the most up-to-date publications and research information, we didn't scrape data from these sources but use them to implement paper/author searching functions in our application. The original paper data took more than 40GB space to store and processed .csv is of near 1GB size.

[1]https://dblp.uni-trier.de/faq/How+can+I+download+the+whole+dblp+dataset
[2]http://papers.nips.cc/

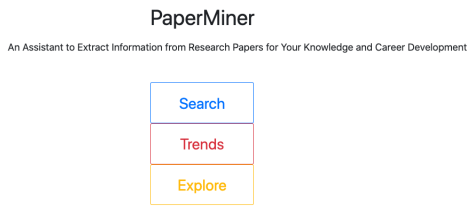
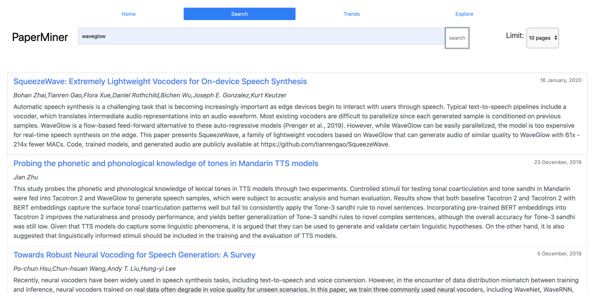

PaperMiner
An Assistant to Extract Information from Research Papers for Your Knowledge and Career Development
Search
Trends
Explore

Figure 1: Home page



SqueezeWave: Extremely Lightweight Vocoders for On-device Speech Synthesis — 16 January, 2020

Bohan Zhai,Tianren Gao,Flora Xue,Daniel Rothchild,Bichen Wu,Joseph E. Gonzalez,Kurt Keutzer

Automatic speech synthesis is a challenging task that is becoming increasingly important as edge devices begin to interact with users through speech. Typical text-to-speech pipelines include a vocoder, which translates intermediate audio representations into an audio waveform. Most existing vocoders are difficult to parallelize since each generated sample is conditioned on previous samples. WaveGlow is a flow-based feed-forward alternative to these auto-regressive models (Prenger et al., 2019). However, while WaveGlow can be easily parallelized, the model is too expensive for real-time speech synthesis on the edge. This paper presents SqueezeWave, a family of lightweight vocoders based on WaveGlow that can generate audio of similar quality to WaveGlow with 61x - 214x fewer MACs. Code, trained models, and generated audio are publicly available at https://github.com/tianrengao/SqueezeWave.

Probing the phonetic and phonological knowledge of tones in Mandarin TTS models — 23 December, 2019

Jian Zhu

This study probes the phonetic and phonological knowledge of lexical tones in TTS models through two experiments. Controlled stimuli for testing tonal coarticulation and tone sandhi in Mandarin were fed into Tacotron 2 and WaveGlow to generate speech samples, which were subject to acoustic analysis and human evaluation. Results show that both baseline Tacotron 2 and Tacotron 2 with BERT embeddings capture the surface tonal coarticulation patterns well but fail to consistently apply the Tone-3 sandhi rule to novel sentences. Incorporating pre-trained BERT embeddings into Tacotron 2 improves the naturalness and prosody performance, and yields better generalization of Tone-3 sandhi rules to novel complex sentences, although the overall accuracy for Tone-3 sandhi was still low. Given that TTS models do capture some linguistic phenomena, it is argued that they can be used to generate and validate certain linguistic hypotheses. On the other hand, it is also suggested that linguistically informed stimuli should be included in the training and the evaluation of TTS models.

Towards Robust Neural Vocoding for Speech Generation: A Survey — 5 December, 2019

Po-chun Hsu,Chun-hsuan Wang,Andy T. Liu,Hung-yi Lee

Recently, neural vocoders have been widely used in speech synthesis tasks, including text-to-speech and voice conversion. However, in the encounter of data distribution mismatch between training and inference, neural vocoders trained on real data often degrade in voice quality for unseen scenarios. In this paper, we train three commonly used neural vocoders, including WaveNet, WaveRNN,

Figure 2: Search Page

3.3 User Interface

Basically, we have four main sections: homepage, search, trends and explore. We have already complete the search page and prvoide some visualization work on other sections.

Homepage. The homepage shows the basic information of PaperMiner, including title, slogan, navigation buttons, see Figure 1. We hope to keep it elegant and informative.

Search. The search page contains the input box, search results within the limit count, the logo and the dropdown list, see Figure 2. The dropdown list provides the page limit options for users, including 10 pages, 20 pages, 50 pages, and 100 pages. For the search engine part, we used a hybrid approach by customizing our own based on available service outputs and self-extracted data like the cluster labels of language model(currently using word2vec[13]) processed paper texts.

Trends. In this page, we present a basic academic data analysis. For instance, we present the paper statistics line chart for different conferences each year,

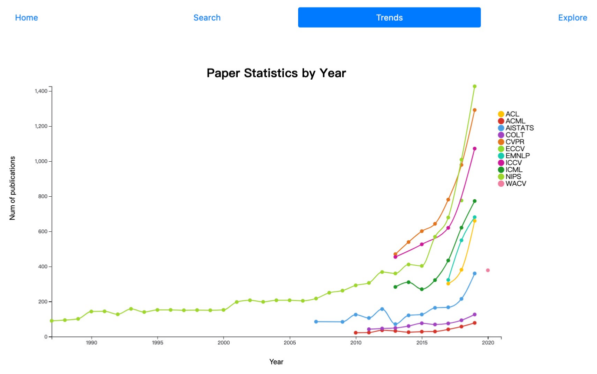


Figure 3: Paper Statistics

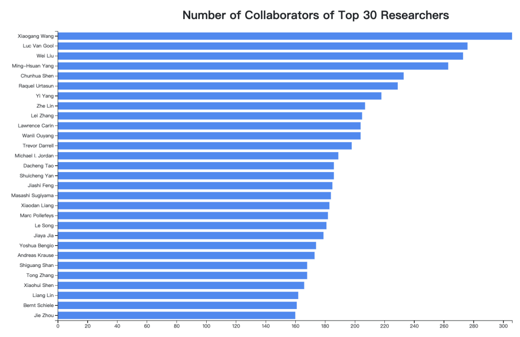
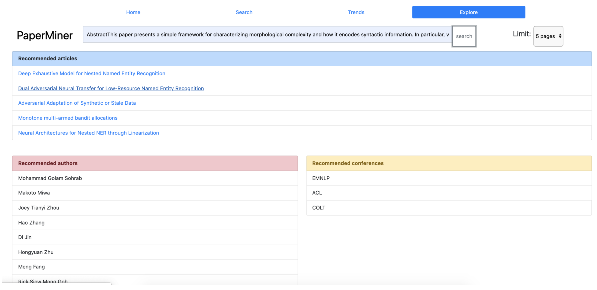


Figure 4: Collaborators Statistics

see Figure 3. PaperMiner can also retrieve the information for collaborators, i.e. the Figure 4 shows the ranking of researchers who have the most collaborators. Users can customize their queries by adding filter labels.

Explore. The explore page gives paper recommendations based on abstract matching, see Figure 5. By embedding our Doc2Vec model, we can present the recommended papers, authors and conferences by simply input a topic you want to know. The explore page helps you find the most relevant papers and their information by matching the abstract. The search query does not need to be the original text, it can be as simple as *"What is Up-propagation Algorithm?"* or *"How does motorized tracking system work?"*.



PaperMiner

AbstractThis paper presents a simple framework for characterizing morphological complexity and how it encodes syntactic information. In particular, w

Recommended articles

- Deep Exhaustive Model for Nested Named Entity Recognition
- Dual Adversarial Neural Transfer for Low-Resource Named Entity Recognition
- Adversarial Adaptation of Synthetic or Stale Data
- Monotone multi-armed bandit allocations
- Neural Architectures for Nested NER through Linearization

Recommended authors

- Mohammad Golam Sohrab
- Makoto Miwa
- Joey Tianyi Zhou
- Hao Zhang
- Di Jin
- Hongyuan Zhu
- Meng Fang
- Rick Siow Mong Goh

Recommended conferences

- EMNLP
- ACL
- COLT

Figure 5: Explore Page

3.4 Algorithm

In PaperMiner, users can retrieve literature and relative information not only by title but also by abstract. We built a text classification with Doc2VeC. Doc2Vec is an extension of well-known Word2Vec, a neural network implementation that learns distributed representations for words, and the K-means algorithm will work together for our current clustering. Doc2Vec was proposed by Quoc Le and Tomas Mikolov in 2014. The learned vector can be used to find the similarity between sentences/ paragraphs/documents by calculating the distance. For labeled data, like papers in our project, we can also use supervised learning to conduct text classification.

Specifically, according to the input searching sentence/keywords, PaperMiner will report the most relevant papers and their information. In simple terms, The algorithm first split the paragraph, delete the stop words and keep the keywords. Then train a Doc2Vec model and use it to find the most similar abstract.

To gain a comparative result, Google's pre-trained Word to Vector Model was loaded to compare with our trained from-scratch Word to Vector Model, and Google's pre-trained Word2Vec model's result was better-off. Take one of the clusters as an example, 'stochastic', 'non-deterministic' and 'probabilistic' are in the same group which shows that a number of stochastic process applications in computer science are discussed in past papers. In the future days, parameters in the model can be tuned for a more accurate outcome and more algorithms could be added for better user experience.

3.5 Connection Between Frontend and Backend

Frontend and backend are connected by HTTP GET requests/responses.

Parameters needed for backend APIs are sent by GET request arguments. Returned data are in json format for frontend's usage.

We wrapped python functions in the backend as APIs using Flask and sent API calls by jQuery's AJAX API in frontend for retrieving data. Currently, we use port 5000 for backend services and port 3000 for frontend nodejs' usage(all backend implemented in python, use nodejs just for faster frontend development) on the same machine, CORS(cross-origin sharing standard) got temporally enabled for easier deployment and demonstration without occupying port 80.

4 EVALUTATION OF PAPERMINER

4.1 Experiment and Evaluation

Besides details on data-processing and text-mining concerning our current focus on data collection and algorithms.

We compared PaperMiner with AMiner and Google Scholar (The most similar platforms) under different tasks. The subjects of the experiments were students including ourselves.

We tested the user experience by concrete scenarios. Some scenarios with significant different performances are listed below.

Scenario 1. A undergraduate seeking PhD program and are interested in Machine Learning but have no idea which college to apply. What information you can give to help make decisions?

Since our platform mainly focuses on the student group, we can find much related information on our platform to solve problems like this. For instance, we give the hot topics with different college professors and their collaborators network. In other words, PaperMiner is more suitable for students' needs.

Scenario 2. The new hot topics and top researchers in Reinforcement Learning after 2018

According to this scenario, Aminer can give us a trend line chart on current hot topics. PaperMiner has the same function. Moreover, PaperMiner can clearly tell the recent hot topics from the flickering word cloud, the keywords with higher frequency will be more likely shown on the word cloud. However, the google scholar is lack of this type of function.

Scenario 3. Accuracy among different Doc2Vec model
In our abstract based recommendation module, we trained the model with increasing *epoch* to find the best epoch. We set the learning rate relative small, and *window* relative high to make sure the optimal parameter would not be stopped at the local optimal solution. By comparing the model before and after the parameter tuning, the model accuracy had increased significantly.

We extracted 10 sentences from our abstract library and rearrange the sentence as the testing data of the experiment. Our tuned model can exactly identify which paper each query sentence comes from, while the untuned model could only reach 20% accuracy.

Scenario 4. User Interface
AMiner has nicer and more colorful plot to improve user experience, which PaperMiner is lacking in this respect. PaperMiner has an elegant and informative interface as Google Scholar, presenting powerful search and academic data analysis functions.

4.2 Possible improvements

The abstract-based recommendation algorithm can be improved further. The Doc2Vec model we implemented in the explore page is pre-trained and is saved locally. The reason is training a model with millions of abstracts can be too time-consuming. Users cannot wait that long for retrieving the result. Furthermore, new papers come out every day, we need to regenerate our model periodically to make sure our search result remains updated. Hence, there's still a large space for improvement.

We put some interesting plots, like word cloud, in PaperMiner. Compared with AMiner, more interesting plots can be put into PaperMiner. To improve the user experience while using our platform, more design standard and philosophy need to be implemented. The UI design of the PaperMiner still has a long way to go.

5 CONCLUSION AND DISCUSSION

This project provides a multi-functional tool, named PaperMiner, for presenting interactive trends and insights in academic papers for the student community in computer science. As scheduled, this application is well-designed with the desired three functions that are search, trends and explore and can mostly help students satisfy their academic needs. More importantly, it has shown that PaperMiner can outperform other similar products in certain respects. The strength of our product is that it is designed specifically for students and provide specific solutions to the problems they are facing in ways that general-use products like Aminer do not. Additionally, the data we collected is comprehensive and most-updated, and algorithm models are trained with high accuracy, so PaperMiner can offer integrated and reliable information to the students community.

5.1 Our Group

The PaperMiner group was comprised of five M.S. students from the Georgia Institute of Technology with different educational backgrounds and this was an asset to success with this project. Our group is well designed to assign the same workload to each team member based on our backgrounds. Zan and Maiqi were responsible for data collection, Jiaji was in charge of frontend development and data visualization, Kai was responsible for academic data analysis, Kaiwen was responsible for the NLP algorithm development. The entire group completed report writing together.

5.2 Future of the Project

PaperMiner is still in an early stage but its algorithm, UI, and connection between frontend and backend are all tested and functional. The project is open source as are all of the libraries we used to build it. Documentations along with the code and this report clearly help understand the whole project, and function modules. The entire code is on the open source website GitHub. Thus, anyone can improve the code or include it into another project freely. We also plan to improve the responsive design of the frontend and modularity of the backend, keep mining current corpus for new findings.

REFERENCES

[1] 2017. arxiv-sanity. http://arxiv-sanity.com/. Accessed: 2020-02-20.

[2] 2017. NIPS Accepted Papers Stats. https://medium.com/machine-learning-in-practice/nips-accepted-papers-stats-26f124843aa0. Accessed: 2020-02-20.

[3] 2017. A web app for ranking computer science departments according to their research output in selective venues. https://github.com/emeryberger/CSrankings. Accessed: 2020-02-21.

[4] 2018. sotawhat. https://github.com/chiphuyen/sotawhat. Accessed: 2020-02-20.

[5] 2019. Browse State-of-the-Art. https://paperswithcode.com/sota/. Accessed: 2020-02-21.

[6] 2020. Microsoft Academic. https://academic.microsoft.com/home. Accessed: 2020-02-21.

[7] Shane Dawson, Dragan Gašević, George Siemens, and Srecko Joksimovic. 2014. Current state and future trends: A citation network analysis of the learning analytics field. In *Proceedings of the fourth international conference on learning analytics and knowledge*. 231–240.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Zhen Guo, Zhongfei Zhang, Shenghuo Zhu, Yun Chi, and Yihong Gong. 2009. Knowledge discovery from citation networks. In *2009 Ninth IEEE international conference on data mining*. IEEE, 800–805.

[10] Norman P Hummon and Patrick Dereian. 1989. Connectivity in a citation network: The development of DNA theory. *Social networks* 11, 1 (1989), 39–63.

[11] Michael Ley. 2009. DBLP: some lessons learned. *Proceedings of the VLDB Endowment* 2, 2 (2009), 1493–1500.

[12] Gerry McKiernan. 2000. arXiv. org: the Los Alamos National Laboratory e-print server. *International Journal on Grey Literature* (2000).

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[14] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The pagerank citation ranking: Bringing order to the web.* Technical Report. Stanford InfoLab.

[15] Jie Tang. 2016. AMiner: Toward understanding big scholar data. In *Proceedings of the ninth ACM international conference on web search and data mining*. 467–467.

[16] Rita Vine. 2006. Google scholar. *Journal of the Medical Library Association* 94, 1 (2006), 97.

[17] Kuansan Wang, Zhihong Shen, Chi-Yuan Huang, Chieh-Han Wu, Darrin Eide, Yuxiao Dong, Junjie Qian, Anshul Kanakia, Alvin Chen, and Richard Rogahn. 2019. A Review of Microsoft Academic Services for Science of Science Studies. *Frontiers in Big Data* 2 (2019), 45.

[18] Stephen L Wright, Michael A Jenkins-Guarnieri, and Jennifer L Murdock. 2013. Career development among first-year college students: College self-efficacy, student persistence, and academic success. *Journal of Career Development* 40, 4 (2013), 292–310.