DISS. ETH NO. 29015

PAPER RETRIEVAL, SUMMARIZATION AND CITATION GENERATION

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

(Dr. sc. ETH Zurich)

presented by

NIANLONG GU

M. Sc., RWTH Aachen University

born on 13.11.1993

citizen of China

accepted on the recommendation of

Prof. Dr. Richard H.R. Hahnloser, examiner

Prof. Dr. Elliott Ash, co-examiner

Dr. Fabio Rinaldi, co-examiner

2023

Abstract

In scientific writing, retrieving, summarizing, and citing relevant papers is necessary but usually time-consuming. Recent research in natural language processing (NLP) has explored the use of neural networks to recommend, summarize and cite papers automatically. However, the following challenges remain before applying these NLP techniques to help authors write scientific articles. First, the rapidly growing volume of available scientific literature has raised the demand for accuracy and efficiency in recommending citations. Second, the length of scientific articles requires high memory efficiency in the summarization model, so long articles do not need to be truncated when summarizing them. Third, the generation of citation sentences needs to be well-controllable to allow authors to direct the generation as they wish. In addition, there is a lack of an integrated system that allows users to search for papers, obtain paper summaries, and get suggested citation sentences to cite them, all in a one-stop shop.

In this thesis, we aim to develop an integrated system for joint paper retrieval, summarization, and citation generation, which consists of the following four parts.

In the first part, to balance speed and accuracy, we propose a two-stage citation recommendation system that first prefetches K candidate papers by embedding-based K-nearest neighbor search and then reranks the prefetched papers with a fine-tuned SciBERT.

In the second part, we develop a reinforcement learning-based sentence extraction model that summarizes a document by iteratively scoring sentences based on the extraction history (e.g., which sentences were selected) and selecting the highest-scoring sentence. Moreover, the lightweight structure allows our model to summarize long scientific articles efficiently and surpass previous state-of-the-art BERT-based extractive summarizers.

In the third part, we propose a controllable citation generation model that users can control by specifying citation attributes. We define the citation attributes as the intent of the citation (e.g., to introduce context or to compare results), the keywords that the user expects to appear in the

citation or the specific sentences in the body of the cited paper that are most relevant to the expected citation sentences.

In the final part, we integrate the subsystems for paper retrieval, summarization, and citation generation into a convenient user interface that displays recommended papers, extracted summaries of recommended papers, and abstractively generated citation sentences that are consistent with the context and selected keywords.

Our work is a step toward applying NLP techniques to help authors write academic papers in real-life scenarios and one of the early attempts at artificial intelligence (AI)-driven scientific inference.

Zusammenfassung

Beim wissenschaftlichen Schreiben ist das Auffinden, Zusammenfassen und Zitieren relevanter Artikel notwendig, aber in der Regel zeitaufwändig. Jüngste Forschungen im Bereich der natürlichen Sprachverarbeitung (NLP) haben den Einsatz neuronaler Netze zur automatischen Empfehlung, Zusammenfassung und Zitierung von Artikeln untersucht. Die Anwendung dieser NLP-Techniken zur Unterstützung von Autoren beim Verfassen wissenschaftlicher Artikel birgt jedoch noch die folgenden Herausforderungen. Erstens hat das schnell wachsende Volumen der verfügbaren wissenschaftlichen Literatur die Nachfrage nach Genauigkeit und Effizienz bei der Empfehlung von Zitaten erhöht. Zweitens erfordert die Länge wissenschaftlicher Artikel eine hohe Speichereffizienz des Zusammenfassungsmodells, so dass lange Artikel bei der Zusammenfassung nicht gekürzt werden müssen. Drittens muss die Generierung von Zitationssätzen gut steuerbar sein, damit die Autoren die Generierung nach ihren Wünschen steuern können. Darüber hinaus fehlt ein integriertes System, das es den Nutzern ermöglicht, in einer einzigen Anlaufstelle nach Artikeln zu suchen, Zusammenfassungen zu erhalten und Zitiervorschläge für diese Artikel zu bekommen.

In dieser Arbeit soll ein integriertes System für die gemeinsame Suche nach Artikeln, die Zusammenfassung und die Erstellung von Zitaten entwickelt werden, das aus den folgenden vier Teilen besteht.

Um ein Gleichgewicht zwischen Geschwindigkeit und Genauigkeit zu erreichen, schlagen wir im ersten Teil ein zweistufiges System für Zitierempfehlungen vor, das zunächst K Kandidatenbeiträge durch eine einbettungsbasierte K-Nächste-Nachbarn-Suche vorfindet und dann die vorgefragten Beiträge mit einem fein abgestimmten SciBERT neu ordnet.

Im zweiten Teil entwickeln wir ein auf Verstärkungslernen basierendes Satz-Extraktionsmodell, das ein Dokument zusammenfasst, indem es iterativ Sätze auf der Grundlage des Extraktionsverlaufs bewertet (z. B. welche Sätze ausgewählt wurden) und den Satz mit der höchsten Bewertung auswählt. Die leichtgewichtige Struktur ermöglicht es unserem Modell, lange wissenschaftliche Artikel effizient zusammenzufassen und

die bisherigen BERT-basierten extraktiven Zusammenfassungen zu übertreffen.

Im dritten Teil schlagen wir ein kontrollierbares Modell für die Generierung von Zitaten vor, das der Benutzer durch die Angabe von Zitierattributen steuern kann. Wir definieren die Zitierattribute als die Absicht des Zitats (z. B. zur Einführung von Kontext oder zum Vergleich von Ergebnissen), die Schlüsselwörter, die der Benutzer im Zitat erwartet, oder die spezifischen Sätze im Textkörper der zitierten Arbeit, die für die erwarteten Zitatsätze am relevantesten sind.

Im letzten Teil integrieren wir die Teilsysteme für das Auffinden von Artikeln, die Zusammenfassung und die Generierung von Zitaten in eine komfortable Benutzeroberfläche, die empfohlene Artikel, extrahierte Zusammenfassungen der empfohlenen Artikel und abstrakt generierte Zitatsätze anzeigt, die mit dem Kontext und den ausgewählten Schlüsselwörtern übereinstimmen.

Unsere Arbeit ist ein Schritt in Richtung der Anwendung von NLP-Techniken zur Unterstützung von Autoren beim Verfassen akademischer Arbeiten in realen Szenarien und einer der ersten Versuche einer von künstlicher Intelligenz (AI) gesteuerten wissenschaftlichen Inferenz.

Acknowledgements

I am grateful to Prof. Dr. Richard Hahnloser for allowing me to work in his group, guiding me in research, and providing support in access to computing resources. During my Ph.D. study, he has provided me with much freedom in exploring research topics, enabling me to learn, think, and gradually understand what it needs to become a junior researcher. I also thank him for his generous help in revising my writing.

I thank Prof. Dr. Elliott Ash and Dr. Fabio Rinaldi, who agreed to be on my thesis committee and provided me with essential feedback on the planning and writing of the thesis.

I thank my fellow Ph.D. students, Dr. Nikola Nikolov, Yingqiang Gao, and Jessica Lam Jia Hong, who have shown great enthusiasm for research in the field of NLP. I have spent many hours discussing and brainstorming with them, which has been very inspiring.

I am grateful to Jonathan Prada, Dr. Andrei Plamada, and Dr. Onur Gökçe, with whom I worked on developing our group's NLP project, where I learned a lot about application-oriented data management, as well as project management and teamwork.

I thank all my colleagues at the NCCR evolving language TTF data science group who have been so kind and supportive.

In addition, I thank everyone in the birdsong group for being so friendly, and we had a great time discussing the study of syllable segmentation in birdsong.

I also thank everyone at INI who makes our institute feel like a family.

Most importantly, I thank my mother, who has supported me since I learned my first Chinese characters as a child. I am also grateful to my wife, Mei, who has accompanied me from Germany to Switzerland for the past five years and has been so supportive and considerate.

Contents

Contents **vii**

1 Introduction **1**
1.1 Background . . . 3
1.2 Problems and Motivation . . . 5
1.2.1 Paper Retrieval . . . 5
1.2.2 Scientific Paper Summarization . . . 6
1.2.3 Citation Generation . . . 8
1.2.4 Integrated Platform for Joint Retrieval, Summarization and Citation Generation . . . 8
1.3 Overview of Contributions . . . 9
1.3.1 Paper Retrieval . . . 10
1.3.2 Scientific Paper Summarization . . . 11
1.3.3 Citation Generation . . . 12
1.3.4 Integrated Platform for Joint Retrieval, Summarization and Citation Generation . . . 13

2 Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking **15**
2.1 Introduction . . . 16
2.2 Related Work . . . 18
2.3 Proposed Dataset . . . 20

2.4 Approach . . . 21
2.4.1 Prefetching Model . . . 21
2.4.2 Reranking Model . . . 23
2.4.3 Loss Function . . . 24
2.5 Experiments . . . 26
2.6 Results and Discussion . . . 28
2.6.1 Prefetching Results . . . 28
2.6.2 Reranking Results . . . 29
2.6.3 Performance of entire Recommendation Pipeline . . . 30
2.7 Conclusion . . . 31

3 MemSum: Extractive Summarization of Long Documents Using Multi-Step Episodic Markov Decision Processes 33
3.1 Introduction . . . 34
3.2 Related Work . . . 36
3.3 Model . . . 37
3.3.1 Policy Gradient Methods . . . 37
3.3.2 Multi-step Episodic MDP Policy . . . 39
3.3.3 Policy Network . . . 39
3.3.4 Training . . . 41
3.4 Experiments . . . 42
3.5 Results and Discussion . . . 44
3.5.1 Results Comparison . . . 45
3.5.2 Ablation Test . . . 48
3.5.3 History Awareness Avoids Redundancy . . . 49
3.5.4 Human Evaluation . . . 50
3.6 Conclusion . . . 52
3.7 Appendices . . . 53
A Computing Hardware . . . 53
B Comparison of Validating and Testing Performance . . . 53
C Summarization Time . . . 53
D Selection of optimal stopping threshold . . . 53
E Creating High-ROUGE Episodes for Training . . . 56

F Padding and Truncation of Sentences and Documents . . 58
G Interactive Web Interface for Human Evaluation 58
H Examples of Extracted Summaries 60
I Reproducibility 60

4 Controllable Citation Text Generation 65
4.1 Introduction 66
4.2 Related Work 68
4.3 Model 69
4.3.1 Conditional Citation Generation Module 69
4.3.2 Attribute Suggestion Module 70
4.4 Dataset Preparation 72
4.4.1 Cited Paper Filtering 72
4.4.2 Citation Sentence Filtering and Train/Test Decoupling . . 73
4.4.3 Parsing Citation Attributes 74
4.5 Experiment 76
4.6 Results and Discussion 77
4.6.1 Results Comparison 78
4.6.2 Controllability of CCG Module 79
4.6.3 Human Evaluation 81
4.7 Conclusion 83
4.8 Appendices 83
A Computing Hardware 83
B Web Interface for Human Evaluation 83

5 SciLit: A Platform for Joint Scientific Literature Discovery, Summarization and Citation Generation 87
5.1 Introduction 88
5.2 SciLit 90
5.2.1 Literature Discovery 90
5.2.2 Extractive Summarization 93
5.2.3 Citation Generation Module 94
5.2.4 Microservice-based Architecture 97

5.3 Evaluation 97
5.3.1 Demonstration 97
5.3.2 Performance 98
5.3.3 Ablation Study 100
5.4 Related Work 100
5.5 Conclusion and Future Work 101
5.6 Appendices 102
A Hardware Information 102
B JSON Schema for Database 102
C Prefetching Indexing Implementation 102
D Joint Retrieval and Citation Generation Examples 106

6 Conclusion and Future Work **109**
6.1 Paper Retrieval 109
6.2 Document Summarization 110
6.3 Controllable Citation Generation 111
6.4 SciLit 112
6.5 Overall Conclusion 112

Bibliography **113**

Chapter 1

Introduction

The process of scientific inference is to make **logical conclusions** based on **observations** and **prior knowledge** (Okasha, 2016). In the case of the discovery of Neptune, scientists observed irregularities in the orbit of Uranus compared to the ideal orbit calculated using Newton's laws of motion and gravity (Valtonen et al., 2016). Based on these observations and knowledge of physics, scientists concluded that an undiscovered object was disturbing the orbit of Uranus, and they successfully predicted the location of this new object, which led directly to the discovery of Neptune. In this scientific inference process, the irregularity of Uranus' orbit is the result of observation, Newton's laws and mathematics are the a priori knowledge, and the hypothesis of the existence of a new planet is the correct inference. The discovery of Neptune was a remarkable success of Newton's law of gravity, showing the power of scientific inference. Since then, an increasing number of scientific breakthroughs have been made using the process of scientific inference. For example, scientists founded quantum physics and relativity from the two "cloud" that obscured physics (Kelvin, 1901).

Despite its power, the scientific inference process is challenging because scientists need to spend much time acquiring the necessary a priori knowledge by reading relevant papers. Moreover, despite the time spent to acquire a priori knowledge, the capacity limitation of the human brain may still result in insufficient a priori knowledge to make reasonable and unbiased inferences. To

address this challenge, in this thesis, we explore whether we can use artificial intelligence (AI) to automate the process of scientific inference.

Having machines automatically perform scientific reasoning is attractive because of the following benefits. First, automated scientific reasoning can help researchers quickly analyze observations and abstract knowledge, thus speeding up scientific research. Second, machines can have an almost infinite amount of "memory" compared to the human brain. If we can find a way to enable a machine to "digest" human knowledge, this machine will have enough a priori knowledge to make unbiased and reasonable scientific inferences. Thus, having machines automatically perform scientific inference is a promising direction. However, it remains an open question how to define interpretable representations of human knowledge that machines can understand and how to model the inference process so that we can train and evaluate inference systems.

A concrete case of scientific inference in scientific writing is **writing the discussion section** of a manuscript. First, it is typical that authors write a "Results" section and then a "Discussion" section. For example, in the PubMed Central Open Access (PMCOA) (of Medicine, 2003) subset that contains papers mainly from the biomedical domain, more than 63% of all papers have a "Results" section that is followed by a "Discussion" section, so it is logically natural to write the discussion on the given results text. Second, when writing the discussion section, authors usually need to compare the experimental results presented in the results section with related literature and make reasonable inferences. In this process, the results section in the written manuscript can be treated as the *observation*, the related literature can be regarded as the *prior knowledge*, and finally, the statements in the discussion section are treated as the *conclusions, deduction or inference*. All three essential components of scientific inference are involved in writing the discussion section. In addition, the results section, the related literature, and the conclusion text can all be represented as natural language, which makes it feasible in theory to automate the discussion writing using natural language processing (NLP) techniques.

However, developing an NLP model to generate the discussion section auto-

matically remains challenging. A significant obstacle comes from the endogenous complexity of the results and discussion sections. For example, the results section in a paper may contain the results of different experiments, each with a different purpose (e.g., testing a different hypothesis). Correspondingly, the discussion section may include a discussion of different results and many related papers. Thus, if we consider the generation of the discussion section as a sequence-to-sequence task, both the length of the input (results section and related papers) and the length of the output (discussion section) are already beyond the capabilities of state-of-the-art (SOTA) seq2seq models. For example, current SOTA text generation models are usually based on a Transformer (Vaswani et al., 2017), which usually cannot input text with more than 512 or 1024 tokens. Therefore, it is necessary to decompose the discussed generation task into finer granularity.

The discussion section consists of a series of sentences, of which we are most interested in the citation sentences that cite and discuss other relevant papers. The reasons for this are as follows. First, citing and discussing relevant works and analyzing experimental results are one of the main tasks of the discussion section. More importantly, writing citation sentences can be seen as a process of scientific inference on a microscopic scale because it also inputs the context in the manuscript and relevant papers as the observation and prior knowledge and makes conclusions (or statements). Therefore, in this thesis, instead of attempting to generate the entire discussion section, we investigate a preliminary task as a first step towards NLP-driven discussion generation: given the context of a manuscript, use NLP techniques to **find** a relevant paper, **summarize** the found paper, and **generate a citation sentence** that cites and discusses this paper in the context of the manuscript.

1.1 Background

Recent research has focused on using natural language processing (NLP) techniques to help researchers retrieve, summarize, and cite scientific papers. First, to help users find relevant papers, Bhagavatula et al. (2018); Ali et al. (2021b,a)

proposed using the global contextual information (title and abstract) of a paper as the query to retrieve relevant papers that users can cite in the body of the query paper. In contrast, He et al. (2010); Ebesu and Fang (2017); Färber et al. (2020); Yang et al. (2019); Jeong et al. (2020); Medić and Snajder (2020) investigated a finer granularity, using local contexts (e.g., short passages with missing citation information) as queries and training the neural network to recommend papers that can be cited in the local contexts.

Second, to help readers effectively capture the main content, neural summarization models have made significant progress in producing compact document summaries. These models have been successfully applied to summarize news/social media texts (Hermann et al., 2015; Zhong et al., 2020; Jia et al., 2020; Lewis et al., 2020), Wikipedia articles (Liu* et al., 2018), and legal documents (Eidelman, 2019). Recent studies have also investigated summarizing a scientific paper into a summary, which can either be extractive or abstractive. Here, the extractive summarization of scientific papers aims to extract a list of sentences from the body text of the paper, such as Zhou et al. (2018); Xiao and Carenini (2019). In contrast, the abstractive summarization aims to generate a short passage by a sequence-to-sequence model, taking body text as input, such as Gidiotis and Tsoumakas (2020); Huang et al. (2021).

Furthermore, to help researchers write citation sentences that cite and discuss relevant papers in the context of the manuscript they are writing, Xing et al. (2020) proposed an automatic citation generation model. This model generates a citation sentence taking two input sources: 1) the local contextual text in the manuscript, i.e., the three sentences before and after the target citation sentence that we want to generate, and 2) the abstract of the target paper that researchers want to cite. Ge et al. (2021); Wang et al. (2022) extended this framework by encoding textual information (e.g., titles, abstracts) and structural information (e.g., citation graph information) to enhance the representation of the cited papers when generating citation sentences. Recent studies also explored the task of generating a related work section given the information of the papers in the reference list (Chen and Zhuge, 2019; Wang et al., 2019; Shah and Barzilay, 2021; Li and Ouyang, 2022).

1.2 Problems and Motivation

While researchers have made great strides in these areas, bringing light to automated literature retrieval, summarization, and citation generation, the following challenges remain when we apply these NLP techniques to help researchers and authors write scientific articles in real-world settings.

1.2.1 Paper Retrieval

The rapidly growing volume of existing scientific literature places greater demands on the retrieval accuracy and efficiency of the paper recommendation system. For example, there are more than 136 million papers in the scientific corpus S2ORC (Lo et al., 2020), 1.7 million articles on the arXiv preprint server (Kaggle, 2022), and 2.7 million papers in PubMed Central Open Access (PMCOA, the commercial use section (of Medicine, 2003)), and these numbers are increasing daily. Although the fast-growing literature databases increase the chance of including papers of interest to researchers, it also raises the difficulty of finding relevant papers in today's era of information overflow. The rapid growth of literature databases has increased the opportunity to include papers of interest to researchers. However, it has also increased the difficulty of retrieving relevant papers because the retrieval system has to find the most relevant papers from an extensive database of hundreds of millions of documents. In addition, this process must be efficient so that users can avoid many delays.

Recommendation models that use graph convolutional networks (GCNs) (Kipf and Welling, 2017; Jeong et al., 2020) or large pre-trained language models such as BERT (Devlin et al., 2019) can achieve high recall of the target citation. However, due to the high computation complexity, they cannot be easily up-scaled to rank millions of candidates. In contrast, embedding-based methods (Kobayashi et al., 2018; Bhagavatula et al., 2018; Gökçe et al., 2020) rank documents based on cosine similarity (or Euclidean distance) between the embedding (a vector representation of text, encoded by text embedding models (Pagliardini et al., 2018; Conneau et al., 2017; Pennington et al., 2014)) of the

query and those of the documents to obtain K nearest neighbors (KNN). Such embedding-based methods are efficient because the document embeddings can be pre-computed, and the KNN search can be sped up with approximate KNN algorithms (Malek Esmaeili et al., 2012; Fu et al., 2019; Indyk and Motwani, 1998) or GPU acceleration (Johnson et al., 2019). However, they are less accurate than BERT-based retrieval methods, as discussed in Guo et al. (2020). Therefore, the trade-off between accuracy and speed must be considered when designing a practical paper recommendation system.

1.2.2 Scientific Paper Summarization

The main challenge in summarizing the scientific literature is the length of the scientific documents. As our study (Gu et al., 2022a) shows, the body of scientific papers from the arXiv dataset (Cohan et al., 2018) consists of an average of 5,206 words and 206 sentences. In contrast, in the commonly-used document summarization benchmark CNN/DM, each document to be summarized contains only 692 words and 35 sentences on average. The extended sequence length of scientific papers requires a summarization model with high memory efficiency so that long documents with hundreds of sentences and thousands of tokens can be summarized.

Transformer-based summarization models have been successfully applied in extractive and abstract summarization tasks. However, such models inherit the sequence length limit of the Transformer (Vaswani et al., 2017) or BERT (encoder part of the Transformer model, Devlin et al. (2019)), typically 512 or 1024 tokens, so this limits the application of Transformer-based models for summarizing long documents such as scientific literature.

For example, to extractively summarize documents in the CNN/DM dataset, MatchSum (Zhong et al., 2020) first used a modified BERT (Liu, 2019) to score all sentences in the document. They then included all possible combinations of two (or three) of the five highest-scoring sentences as candidate summaries. Then, they encoded the documents and candidate summaries as embeddings using another fine-tuned BERT and ranked the candidate summaries based

on the cosine similarity between the document embeddings and the candidate summary embeddings. Under such a setup, when the length of the document exceeds the length limit of BERT (typically 512 tokens), MatchSum needs to truncate it when scoring sentences and computing the document embedding. As a result, sentences located in the truncated part of the document will not be selected for the summary, which may affect the summarization performance. We showed that MatchSum performs worse in summarizing PubMed papers due to truncating the original document (Gu et al., 2022a).

The length limit of the Transformer and the BERT model results from the high memory complexity of the computation of attentions:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V} \tag{1.1}$$

$\mathbf{Q}$,$\mathbf{K}$,$\mathbf{V} \in \mathcal{R}^{n \times d_k}$ represent the query, key, and value matrices in each attention layer for all n tokens in the sequence, and d_k represents the hidden dimension. Thus, the memory complexity is $\mathcal{O}(n^2)$ (Huang et al., 2021), which means that the memory footprint will increase quadratically as the sequence length increases. Huang et al. (2021) proposed an efficient Transformer with a modified attention computation operation that reduces memory complexity by a linear factor. They used this model to abstractively summarize long documents, including scientific articles and government reports. However, this modified Transformer still has many trainable parameters, making it difficult to train from scratch and relatively slow in the inference process.

In this work, we study extractive summarizers because they typically produce more reliable summaries in terms of syntax and content than abstractive summarizers (Liu and Lapata, 2019a; Luo et al., 2019; Liao et al., 2020). Our goal is to propose an extractive summarization model with the following properties: 1) It extractively summarizes scientific documents by selecting sentences from the entire body text without truncating the body into 512 or 1024 tokens. 2) It has a lightweight architecture that is memory efficient and fast in summarizing documents in real-time.

1.2.3 Citation Generation

The goal of citation generation is to generate a citation sentence to cite and discuss the paper in the context of the manuscript. This task has been modeled as an abstractive summarization problem. For example, Xing et al. (2020) proposed a sequence-to-sequence model that inputs the contextual sentences in the manuscript and the abstract of the cited paper and outputs a citation sentence. However, this setup oversimplifies the actual situation of citing a paper, ignoring the fact that the way a paper is cited can vary depending on various conditional attributes (Dathathri et al., 2020), such as 1) the citation intent (Cohan et al., 2019), e.g., introducing the background, describing the method, or comparing the results, 2) the keywords that motivate the citation of a particular paper, and 3) relevant sentences in the body of the cited paper that may contain the details that the author wants to cite. We argue that the citation generation model trained in this simplified setting cannot adjust the generated citations according to the conditions specified by the user, which may limit its application to aid authors in citing papers in real-world scenarios.

Instead of only highlighting a high degree of automation, as in previous studies (Xing et al., 2020; Ge et al., 2021; Wang et al., 2022), our aim is to develop a more controllable citation generation system. This system can produce diverse citation sentences when users provide different attributes to the citation sentence they expect the model to generate. The better controllability of our citation generation system allows users to flexibly fine-tune the generated citation sentences, thus making it more likely that appropriate citation sentences will be generated to meet the author's citation requirements.

1.2.4 Integrated Platform for Joint Retrieval, Summarization and Citation Generation

It is challenging to benefit researchers' scientific writing processes when NLP-driven functions are hosted separately. For example, when writing a scientific paper, researchers must refer to a literature search engine when they need to find relevant papers. When finding papers of interest and want to obtain

highlights, they need to call a document summarization algorithm on another platform. Finally, when generating citation sentences, they may need to upload their manuscript texts and cited papers to another platform to obtain the cited citation sentences. Such a cumbersome process can prevent users from using these newly proposed tools.

Therefore, this work aims to build an integrated platform that helps authors search, read and write scientific literature in an immersive environment. We will integrate our developed literature search module, extractive summarization module, and citation text generation module into the platform to help users search for relevant papers, extract highlights and generate reference text in a one-stop shop. It can also be easily extended with other NLP-driven features.

1.3 Overview of Contributions

The following thesis is cumulative and consists of four open-access publications corresponding to four sub-topics of our work:

1. **Paper Retrieval**:

   Gu, Nianlong, Yingqiang Gao, and Richard HR Hahnloser. "Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking." European Conference on Information Retrieval. Springer, Cham, 2022.

2. **Scientific Paper Summarization**:

   Gu, Nianlong, Elliott Ash, and Richard Hahnloser. "MemSum: Extractive Summarization of Long Documents Using Multi-Step Episodic Markov Decision Processes." Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022.

3. **Citation Generation**:

   Gu, Nianlong and Richard Hahnloser. "Controllable Citation Text Generation." arXiv preprint arXiv:2211.07066 (2022).

4. **Integrated Platform for Joint Retrieval, Summarization and Citation Generation**:

   *Gu, Nianlong and Richard Hahnloser. "*SciLit*: A Platform for Joint Scientific Literature Discovery, Summarization and Citation Generation." (to be submitted)*

We summarize our contribution in these work (and my personal contribution) as follows.

1.3.1 Paper Retrieval

In this work, we proposed a competitive paper retrieval system built on a two-stage pipeline for prefetching and reranking, respectively. In the prefetching phase, we designed a hierarchical attentional text encoder (HAtten) that can efficiently encode scientific literature into a vector that can be used for embedding-based K-nearest neighbor search. Using HAtten, we can efficiently prefetch a small number (e.g., 100) of candidate papers. We then fine-tuned a SciBERT model that takes as input the concatenation of query text and candidate paper text and outputs a score representing the correlation between the query and the paper. We then used this score to rerank the prefetched candidate papers. In the evaluation of the entire pipeline, we demonstrated a balance between speed and accuracy. We also released our code and a large dataset of scientific papers for further study.

Personal contributions: In this work, I created a new dataset by parsing the S2ORC corpus, independently designed and implemented the hierarchical attentional text encoder, and fine-tuned the SciBERT model for reranking. I explored the impact of different training losses on recommendation performance. I proposed the use of a combination of positive and negative mining strategies to extract effective triplet pairs, which helped speed up the training convergence of the hierarchical attention text encoder. I led the writing of the paper and wrote the first draft.

Collaborations: At the beginning of the work, I discussed with Dr. Nikola Nikolov the design of a two-stage pipeline for prefetching and reranking. Af-

ter I completed the first version of the paper, Yingqiang Gao and Prof. Richard Hahnloser were involved in revising the paper. In the second version of the paper, which was the final version accepted in ECIR 2022, Prof. Richard Hahnloser was involved in revising the paper.

1.3.2 Scientific Paper Summarization

In this work, we proposed MemSum, a model that considers extractive summarization as a multi-step episodic Markov decision process. Given a document with a list of sentences, we modeled extractive summarization as a multi-step process. We iteratively score the sentences in the document and select the sentence with the highest score into the summary. Our model recalculates the scores of all remaining unselected sentences at each sentence selection step based on three state information sources: 1) the text of the sentence, 2) the global context of the sentence in the document, and 3) the information about the extraction history, including which sentences have been extracted and which sentences are remaining.

Our results showed that extraction-history awareness allowed our model to extract more compact summaries than models without history awareness and behave more robustly to redundancies in documents. In addition, we encoded the state information using a lightweight model based on Bi-directional LSTMs and a shallow attention network, which enabled us to summarize long scientific documents in a memory-efficient way. Our human evaluation results show that the extractive summaries produced by MemSum are of higher quality than those of a competitive approach, especially with fewer redundancies.

Personal contributions: In this work, I conducted the design and implementation of MemSum and trained and tested the model on various datasets, including PubMed, arXiv, and GovReport. I also implemented a web interface for the human evaluation experiments conducted in this paper. I wrote the first draft and led the revision process until it was accepted.

Collaborations: Prof. Richard Hahnloser and Prof. Elliott Ash participated in the review and rebuttal phase of the paper during the ACL 2021, AAAI 2022,

and ACL 2022 (accepted) reviews. We exchanged ideas about improving the model and what experiments to add when revising the paper.

1.3.3 Citation Generation

In this work, we proposed a controllable citation generation pipeline consisting of two modules: 1) the Conditional Citation Generation (CCG) module and 2) the citation attribute suggestion module.

We designed the CCG module to take 1) the contextual texts in the manuscript and the title and abstract of the cited paper as contextual inputs and 2) the citation attributes of target citation sentences, including citation intent, relevant keywords, and relevant sentences, as conditional inputs. In addition, we proposed a SciBERT-based attribute suggestion module that can suggest candidate keywords, sentences, and the most likely citation intent. With the attribute suggestion module, users can effectively select the suggested attributes (instead of composing them from scratch) to control the generation process of the CCG module.

The experimental results showed that our CCG module has good controllability over the various citation attributes provided, and our attribute suggestion module effectively suggests relevant citation attributes.

Personal contributions: I created a new dataset for controllable citation generation by parsing the S2ORC corpus. In this newly proposed dataset, each instance contains a thoroughly cleaned citation sentence and complete information about the citing paper (where the citation sentence comes from) and the cited paper (the paper cited by the citation sentence). This dataset can be used for future studies on controlled citation generation.

Inspired by research on controlled text generation (Keskar et al., 2019; Dathathri et al., 2020), I designed and implemented the controlled citation generation module based on BART-large (Lewis et al., 2020). Furthermore, by modifying and fine-tuning the SciBERT model, I designed the citation attribute suggestion module consisting of a keyword extractor, a sentence extractor, and a citation intent predictor. For comparison with the baseline, PTGEN-Cross

(Xing et al., 2020), since the authors did not release the original code, I reimplemented the citation generation model proposed in Xing et al. (2020). In addition, I designed and implemented a web platform using React JS to conduct human evaluation experiments to test the controllability and user satisfaction of our controllable citation generation pipeline. I led the writing of the paper and wrote the first draft.

Collaborations: The idea of designing a controlled citation generation system originated from the discussion with Prof. Richard Hahnloser about how users could flexibly guide the generated citation text by providing some hints, such as keywords. Prof. Richard Hahnloser also contributed to the draft's revision and provided critical feedback on the design of the human evaluation platform.

1.3.4 Integrated Platform for Joint Retrieval, Summarization and Citation Generation

In this system demonstration work, we demonstrated SciLit, a platform for one-stop searching, summarizing and citing scientific papers. This platform consists of an NLP backend and a frontend. We implemented in the backend the algorithms for paper retrieval, summarization, and citation generation, based on our work in this thesis. We set up a user-friendly web page in the frontend using React JS.

We evaluated SciLit in the tasks of scientific literature retrieval, paper summarization, and citation sentence generation, and showcased the generation of a related-work paragraph. We also open-sourced our database and the algorithms used to build our system.

Personal contributions:

In this work, I designed the overall architecture of the system and then implemented each module independently. I parsed more than 136 million scientific papers from S2ORC and built a database of scientific papers using SQLite (Hipp, 2000). To efficiently retrieve papers from this extensive database, I

developed a two-stage prefetching-ranking search engine following our work (Gu et al., 2022b). In the prefetching phase, I used an unsupervised sentence embedding model, Sent2Vec (Pagliardini et al., 2018), to compute document embeddings and query embeddings, which were then used for embedding-based K-nearest neighbor search. In the meantime, I built an inverted index on the database to allow the filtering of papers for a given keyword (keyword Boolean filtering). During the reordering phase, I fine-tuned the SciBERT model based on our work (Gu et al., 2022b). Moreover, based on our work on long document extractive summarization and controllable citation text generation that are introdcued in this thesis, I implemented paper summarization and citation generation functions and set up the corresponding services in our NLP backend.

In addition, I independently implemented the front-end web page and wrote the first draft of the system demonstration paper.

Collaborations: I worked with Dr. Onur Gökçe to design a prototype inverted index for the paper search module. At that time, the inverted index was designed to be placed in RAM and could not be used to create indexes for a large number of scientific papers. This part of the work has been published in our previous work (Gökçe et al., 2020). Instead, in this work, I improved the inverted index by building an on-disk hash map using sqlitedict, which allowed us to index more than 136 million papers without much memory.

In collaboration with Dr. Andrei Plamada and Jonathan Prada, we built a JSON schema that defines the data structure of how scientific papers should be stored in our paper database. Prof. Richard Hahnloser supervised the entire design and implementation phase and participated in revising the paper.

Chapter 2

Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking

Originally published as: Gu, Nianlong, Yingqiang Gao, and Richard HR Hahnloser. "Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking." European Conference on Information Retrieval. Springer, Cham, 2022.

Abstract

The goal of local citation recommendation is to recommend a missing reference from the local citation context and optionally also from the global context. To balance the tradeoff between speed and accuracy of citation recommendation in the context of a large-scale paper database, a viable approach is to first prefetch a limited number of relevant documents using efficient ranking methods and then to perform a fine-grained reranking using more sophisticated models. In that vein, BM25 has been found to be a tough-to-beat approach to prefetching, which is why recent work has focused mainly on the reranking

step. Even so, we explore prefetching with nearest neighbor search among text embeddings constructed by a hierarchical attention network. When coupled with a SciBERT reranker fine-tuned on local citation recommendation tasks, our hierarchical Attention encoder (HAtten) achieves high prefetch recall for a given number of candidates to be reranked. Consequently, our reranker requires fewer prefetch candidates to rerank, yet still achieves state-of-the-art performance on various local citation recommendation datasets such as ACL-200, FullTextPeerRead, RefSeer, and arXiv.

keywords: Local citation recommendation; Hierarchical attention; Document reranking

2.1 Introduction

Literature discovery, such as finding relevant scientific articles, remains challenging in today's age of information overflow, largely arising from the exponential growth in both the publication record (Hunter and Cohen, 2006) and the underlying vocabulary (Herdan, 1960). Assistance to literature discovery can be provided with automatic citation recommendation, whereby a query text without citation serves as the input to a recommendation system and a paper worth citing as its output (Färber and Jatowt, 2020).

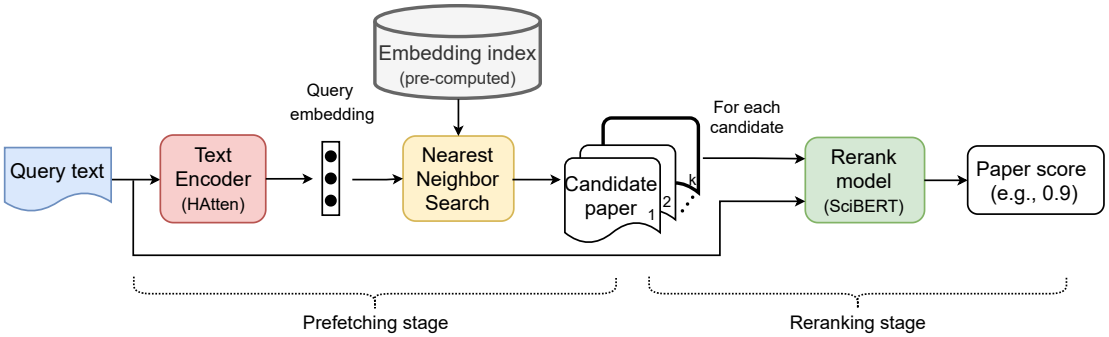


Figure 2.1: Overview of our two-stage local citation recommendation pipeline.

Citation recommendation can be dealt with either as a global retrieval problem (Strohman et al., 2007; Nallapati et al., 2008; Bhagavatula et al., 2018) or as a

local one (He et al., 2010; Huang et al., 2012; Jeong et al., 2020). In global citation recommendation, the query text is composed of the title and the abstract of a source paper (Bhagavatula et al., 2018). In contrast, in local citation recommendation, the query consist of two sources of contexts (He et al., 2010; Medić and Snajder, 2020): 1) the text surrounding the citation placeholder with the information of the cited paper removed (the **local context**); and 2) the title and abstract of the citing paper as the **global context**. The aim of local citation recommendation is to find the missing paper cited at the placeholder of the local context. In this paper we focus on local citation recommendation.

It is important for a local citation recommendation system to maintain a balance between accuracy (e.g., recall of the target paper among the top K recommended papers) and speed in order to operate efficiently on a large database containing millions of scientific papers. The speed-accuracy tradeoff can be flexibly dealt with using a two-step prefetching-reranking strategy: 1) A fast prefetching model first retrieves a set of candidate papers from the database; 2) a more sophisticated model then performs a fine-grained analysis of scoring candidate papers and reordering them to result in a ranked list of recommendations. In many recent studies (Medić and Snajder, 2020; Dai et al., 2020; Ebesu and Fang, 2017; Livne et al., 2014), either (TF-IDF) (Ramos et al., 2003) or BM25 (Robertson and Zaragoza, 2009) were used as the prefetching algorithm, which were neither fine-tuned nor taken into consideration when evaluating the recommendation performance.

In this paper, we propose a novel two-stage local citation recommendation system (Figure 2.1). In the prefetching stage, we make use of an embedding-based paper retrieval system, in which a siamese text encoder first pre-computes a vector-based embedding for each paper in the database. The query text is then mapped into the same embedding space to retrieve the K nearest neighbors of the query vector. To encode queries and papers of various lengths in a memory-efficient way, we design a two-layer Hierarchical Attention-based text encoder (HAtten) that first computes paragraph embeddings and then computes from the paragraph embeddings the query and document embeddings using a self-attention mechanism (Vaswani et al., 2017). In the reranking

step, we fine-tune the SciBERT (Beltagy et al., 2019) to rerank the candidates retrieved by the HAtten prefetching model.

In addition, to cope with the scarceness of large-scale training datasets in many domains, we construct a novel dataset that we distilled from 1.7 million arXiv papers. The dataset consist of 3.2 million local citation sentences along with the title and the abstract of both the citing and the cited papers. Extensive experiments on the arXiv dataset as well as on previous datasets including ACL-200 (Medić and Snajder, 2020), RefSeer (Medić and Snajder, 2020; Ebesu and Fang, 2017), and FullTextPeerRead (Jeong et al., 2020) show that our local citation recommendation system performs better on both prefetching and reranking than the baseline and requires fewer prefetched candidates in the reranking step thanks to higher recall of our prefetching system, which indicates that our system strikes a better speed-accuracy balance.

In total, our main contributions are summarized as follows:

- We propose a competitive retrieval system consisting of a hierarchical-attention text encoder and a fine-tuned SciBERT reranker.
- In evaluations of the whole pipeline, we demonstrate a well-balanced tradeoff between speed and accuracy.
- We release our code and a large-scale scientific paper dataset[1] for training and evaluation of production-level local citation recommendation systems.

2.2 Related Work

Local citation recommendation was previously addressed in He et al. (He et al., 2010) in which a non-parametric probabilistic model was proposed to model the relevance between the query and each candidate citation. In recent years, embedding-based approaches (Kobayashi et al., 2018; Gökçe et al., 2020) have been proposed to more flexibly capture the resemblance between the

[1]Our code and data are available at https://github.com/nianlonggu/Local-Citation-Recommendation.

query and the target according to the cosine distance or the Euclidean distance between their embeddings. Jeong et al. (Jeong et al., 2020) proposed a BERT-GCN model in which they used Graph Convolutional Networks (Kipf and Welling, 2017) (GCN) and BERT (Devlin et al., 2019) to compute for each paper embeddings of the citation graph and the query context, which they fed into a feed-forward network to estimate relevance. The BERT-GCN model was evaluated on small datasets of only thousands of papers, partly due to the high cost of computing the GCN, which limited its scalability for recommending citations from large paper databases. Although recent studies (Medić and Snajder, 2020; Dai et al., 2020; Ebesu and Fang, 2017; Livne et al., 2014) adopted the prefetching-reranking strategy to improve the scalability, the prefetch part (BM25 or TF-IDF) only served for creating datasets for training and evaluating the reranking model, since the target cited paper was added manually if it was not retrieved by the prefetch model, i.e. the recall of the target among the candidate papers was set to 1. Therefore, these recommendation systems were evaluated in an artificial situation with an ideal prefetching model that in reality does not exist.

Supervised methods for citation recommendation rely on the availability of numerous labeled data for training. It is challenging to assemble a dataset for local citation recommendation due to the need of parsing the full text of papers to extract the local contexts and finding citations that are also available in the dataset, which eliminates a large bulk of data. Therefore, existing datasets on local citation recommendation are usually limited in size. For example, the ACL-200 (Medić and Snajder, 2020) and the FullTextPeerRead (Jeong et al., 2020) contain only thousands of papers. One of the largest datasets is RefSeer used in Medić and Šnajder (Medić and Snajder, 2020), which contains 0.6 million papers in total, but this dataset is not up-to-date as it only contains papers prior to 2015. Although unarXive (Saier and Färber, 2020), a large dataset for citation recommendation, exists, this dataset does not meet the needs of our task because: 1) papers in unarXive are not parsed in a structured manner. For example, the abstract is not separated from the full text, which makes it difficult to construct a global context in our experiments; 2) the citation context

is usually a single sentence containing a citation marker, even if the sentence does not contain sufficient contextual information, e.g., "For details, see [#]". These caveats motivate the creation of a novel dataset of high quality.

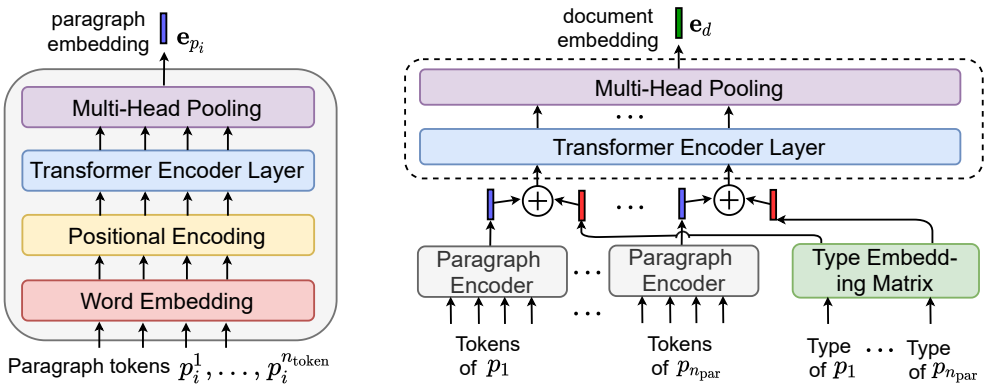
2.3 Proposed Dataset

We create a new dataset for local citation recommendation using arXiv papers contained in S2ORC (Lo et al., 2020), a large-scale scientific paper corpus. Each paper in S2ORC has an identifier of the paper source, such as arXiv or PubMed. Using this identifier, we first obtain all arXiv papers with available titles and abstracts. The title and abstract of each paper are required because they are used as the global context of a query from that paper or as a representation of the paper's content when the latter is a candidate to be ranked. From the papers we then extract the local contexts by parsing those papers for which the full text is available: For each reference in the full text, if the cited paper is also available in the arXiv paper database, we replace the reference marker such as "[#]" or "XXX et al." with a special token such as "CIT", and collect 200 characters surrounding the replaced citation marker as the local context. Note that we "cut off" a word if it lied on the 200-character boundary, following the setting of the ACL-200 and the RefSeer datasets proposed in Medić and Šnajder (Medić and Snajder, 2020).

Dataset	Number of local contexts			Number of papers	publication years
	Train	Val	Test		
ACL-200	30,390	9,381	9,585	19,776	2009 – 2015
FullTextPeerRead	9,363	492	6,814	4,837	2007 – 2017
RefSeer	3,521,582	124,911	126,593	624,957	– 2014
arXiv (Ours)	2,988,030	112,779	104,401	1,661,201	1991 – 2020

Table 2.1: Statistics of the datasets for local citation recommendation.

Table 2.1 shows the statistics of the created arXiv dataset and the comparison with existing datasets used in this paper. As the most recent contexts available in the arXiv dataset is from April 2020, we use the contexts from 1991 to 2019 as the training set, the contexts from January 2020 to February 2020 as the



(a) Paragraph Encoder. **(b)** Document Encoder.

Figure 2.2: The Hierarchical-Attention text encoder (HAtten) used in the prefetching step is composed of a paragraph encoder (a) and a document encoder (b).

validating set, and the contexts from March 2020 to April 2020 as the test set. The sizes of the arXiv training, validating, and testing sets are comparable to RefSeer, one of the largest existing datasets, whereas our arXiv dataset contains a much larger number of papers, and there are more recently published papers available in the arXiv dataset. These features make the arXiv dataset a more challenging and up-to-date test bench.

2.4 Approach

Our two-stage telescope citation recommendation system is similar to that of Bhagavatula et al. (Bhagavatula et al., 2018), composed of a fast **prefetching** model and a slower **reranking** model.

2.4.1 Prefetching Model

The prefetching model scores and ranks all papers in the database to fetch a rough initial subset of candidates. We designed a representation-focused ranking model (Guo et al., 2020) that computes a query embedding for each input query and ranks each candidate document according to the cosine similarity between the query embedding and the pre-computed document embedding.

The core of the prefetching model is a light-weight text encoder that efficiently computes the embeddings of queries and candidate documents. As shown in Figure 2.2, the encoder processes each document or query in a two-level hierarchy, consisting of two components: a paragraph encoder and a document encoder.

Paragraph Encoder

For each paragraph p_i in the document, the paragraph encoder (Figure 2.2a) takes as input the token sequence $p_i = [w_1, \ldots, w_{n_\mathrm{i}}]$ composed of n_i tokens (words) to output the paragraph embedding $\boldsymbol{e}_{p_i}$ as a single vector. In order to incorporate positional information of the tokens, the paragraph encoder makes use of positional encoding. Contextual information is encoded with a single transformer encoder layer following the configuration in Vaswani et al. (Vaswani et al., 2017), Figure 2.2a. To obtain a single fixed-size embedding $\boldsymbol{e}_p$ from a variably sized paragraph, the paragraph encoder processes the output of the transformer encoder layer with a multi-head pooling layer (Liu and Lapata, 2019a) with trainable weights. Let $x_k \in \mathbb{R}^d$ be the output of the transformer encoder layer for token w_k in a paragraph p_i. For each head $j \in \{1, \ldots, n_\mathrm{head}\}$ in the multi-head pooling layer, we first compute a value vector $v_k^j \in \mathbb{R}^{d/n_\mathrm{head}}$ as well as an attention score $\hat{a}_k^j \in \mathbb{R}$ associated with that value vector:

$$v_k^j = \mathrm{Linear}_v^j(x_k), \quad a_k^j = \mathrm{Linear}_a^j(x_k), \quad \hat{a}_k^j = \frac{\exp a_k^j}{\sum_{m=1}^{n_\mathrm{token}} \exp a_m^j}, \tag{2.1}$$

where $\mathrm{Linear}()$ denotes a trainable linear transformation. The weighted value vector $\hat{v}^j$ then results from the sum across all value vectors weighed by their corresponding attention scores: $\hat{v}^j = \sum_{m=1}^{n_\mathrm{par}} \hat{a}_m^j v_m^j$. The final paragraph embedding $\boldsymbol{e}_p$ is constructed from the weighted value vectors $\hat{v}^j$ of all heads by a ReLU activation (Nair and Hinton, 2010) followed by a linear transformation:

$$\boldsymbol{e}_p = \mathrm{Linear}_p(\mathrm{ReLU}(\mathrm{Concat}(\hat{v}^1, \ldots, \hat{v}^{n_\mathrm{head}}))). \tag{2.2}$$

Document Encoder

In order to encode documents with two fields given by the title and the abstract, or to encode queries given by three fields: the local context, the title, and the abstract of the citing paper, we treat each field (local context, title, and abstract) as a paragraph. For a document of n_{par} paragraphs $d = [p_1, \ldots, p_{n_{\text{par}}}]$, we first compute the embeddings of all paragraphs p_i.

Not all fields and paragraphs are treated equally in our document encoder. To allow the document encoder to distinguish between fields, we introduce a *paragraph type* variable, which refers to the field type from which the paragraph originates. We distinguish between three paragraph types: the title, the abstract, and the local context. Each type is associated with a learnable type embedding that has the same dimension as the paragraph embedding. Inspired by the BERT model (Devlin et al., 2019), we produce a type-aware paragraph embedding by adding the type embedding of the given paragraph to the corresponding paragraph embedding (Figure 2.2b). All type-aware paragraph embeddings are then fed into a transformer encoder layer followed by a multi-head pooling layer (of identical structures as the ones in the paragraph encoder), which then results in the final document embedding $\boldsymbol{e}_d$.

Prefetched document candidates

The prefetched document candidates are found by identifying the K nearest document embeddings to the query embedding in terms of cosine similarity. The ranking is performed using a brute-force nearest neighbor search among all document embeddings as shown in Figure 2.1.

2.4.2 Reranking Model

The reranking model performs a fine-grained comparison between a query q (consisting of a local and a global context) and each prefetched document candidate (its title and the abstract). The relevance scores of the candidates constitute the final output of our model. We design a reranker based on SciBERT (Beltagy et al., 2019), which is a BERT model (Devlin et al., 2019) trained on a

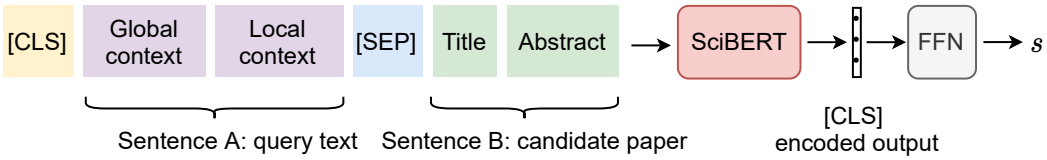


Figure 2.3: Structure of our SciBERT Reranker.

large-scale corpus of scientific articles. The input of the SciBERT reranker has the following format: "[CLS] Sentence A [SEP] Sentence B", where sentence A is the concatenation of the global context (title and abstract of the citing paper) and the local context of the query, and sentence B is the concatenation of the title and the abstract of the candidate paper to be scored, Figure 2.3. The SciBERT-encoded vector for the "[CLS]" token is then fed into a feed-forward network that outputs the relevance score $s \in [0,1]$ provided via a sigmoid function.

2.4.3 Loss Function

We use a triplet loss both to train our HAtten text encoder for prefetching and to finetune the SciBERT reranker. The triplet loss is based on the similarity $s(q,d)$ between the query q and a document d. For the prefetching step, $s(q,d)$ is given by the cosine similarity between the query embedding v_q and the document embedding v_d, both computed with the HAtten encoder. For the reranking step, $s(q,d)$ is given by the relevance score computed by the SciBERT reranker. In order to maximize the relevance score between the query q and the cited document d_+ (the positive pair (q,d_+)) and to minimize the score between q and any non-cited document d_- (a negative pair (q,d_-)), we minimize the triplet loss:

$$\mathcal{L} = \max[s(q,d_-) - s(q,d_+) + m, 0] \tag{2.3}$$

where the margin $m > 0$ sets the span over which the loss is sensitive to the similarity of negative pairs.

For fast convergence during training, it is important to select effective triplets for which $\mathcal{L}$ in Equation (2.3) is non-zero (Schroff et al., 2015), which is par-

ticularly relevant for the prefetching model, since for each query there is only a single positive document but millions of negative documents (e.g., on the arXiv dataset). Therefore, we employ negative and positive mining strategies to train our HAtten encoder, described as follows.

Negative mining Given a query q, we use HAtten's current checkpoint to prefetch the top K_n candidates excluding the cited paper. The HAtten embedding of these prefetched non-cited candidates have high cosine similarity to the HAtten embedding of the query. To increase the similarity between the query and the cited paper while suppressing the similarity between the query and these non-cited candidates, we use the cited paper as the positive document and select the negative document from these K_n overly similar candidates.

Positive mining Among the prefetched non-cited candidates, the documents with objectively high textual similarity (e.g. measured by word overlapping, such as the Jaccard index (Bhagavatula et al., 2018)) to the query were considered relevant to the query, even if they were not cited. These textually relevant candidate documents should have a higher cosine similarity to the query than randomly selected documents. Therefore, in parallel with the negative mining strategy, we also select positive documents from the set of textually relevant candidates and select negative documents by random sampling from the entire dataset.

The checkpoint of the HAtten model is updated every N_{iter} training iterations, at which point the prefetched non-cited and the textually relevant candidates for negative and positive mining are updated as well.

In contrast, when fine-tuning SciBERT for reranking, the reranker only needs to rerank the top K_r prefetched candidates. This allows for a simpler triplet mining strategy, which is to select the cited paper as the positive document and randomly selecting a prefetched non-cited papers as the negative document.

2.5 Experiments

Implementation Details In the prefetching step, we used as word embeddings of the HAtten text encoder the pre-trained 200-dimensional GloVe embeddings (Pennington et al., 2014), which were kept fixed during training. There are 64 queries in a mini-batch, each of which was accompanied by 1 cited paper, 4 non-cited papers randomly sampled from the top $K_n = 100$ prefetched candidates, and 1 randomly sampled paper from the whole database, which allow us to do negative and positive mining with the mini-batch as described in Section 2.4.3. The HAtten's checkpoint was updated every $N_{\text{iter}} = 5000$ training iterations.

In the reranking step, we initialized the SciBERT reranker with the pretrained model provided in Beltagy et al. (Beltagy et al., 2019). The feed-forward network in Figure 2.3 consisting of a single linear layer was randomly initialized. Within a mini-batch there was 1 query, 1 cited paper (positive sample), and 62 documents (negative samples) randomly sampled from the top $K_r = 2000$ prefetched non-cited documents. In the triplet loss function the margin m was set to 0.1.

We used the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In the prefetching step, the learning rate was set to $\alpha = 1e^{-4}$ and the weight decay to $1e^{-5}$, while in the reranking step these were set to $1e^{-5}$ and to $1e^{-2}$ for fine-tuning SciBERT, respectively. The models were trained on eight NVIDIA GeForce RTX 2080 Ti 11GB GPUs and tested on two Quadro RTX 8000 GPUs.

Evaluation Metrics We evaluated the recommendation performance using the Mean Reciprocal Rank (MRR) (Voorhees, 1999) and the Recall@K (R@K for short), consistent with previous work (Medić and Snajder, 2020; Färber and Sampath, 2020; Jeong et al., 2020). The MRR measures the reciprocal rank of the actually cited paper among the recommended candidates, averaged over multiple queries. The R@K evaluates the percentage of the cited paper appearing in the top K recommendations.

Baselines In the prefetching step, we compare our HAtten with the following

baselines: BM25, Sent2vec (Pagliardini et al., 2018), and NNSelect (Bhagavatula et al., 2018). BM25 was used as the prefetching method in previous works (Medić and Snajder, 2020; Ebesu and Fang, 2017; Livne et al., 2014). Sent2vec is an unsupervised text encoder which computes a text embedding by averaging the embeddings of all words in the text. We use the 600-dim Sent2vec pretrained on Wikipedia. NNSelect (Bhagavatula et al., 2018) computes text embeddings also by averaging, and the trainable parameters are the magnitudes of word embeddings that we trained on each dataset using the same training configuration as our HAtten model.

In the reranking step, we compare our fine-tuned SciBERT reranker with the following baselines: 1) a Neural Citation Network (NCN) with an encoder-decoder architecture (Ebesu and Fang, 2017; Färber et al., 2020); 2) DualEnh and DualCon (Medić and Snajder, 2020) that score each candidate using both semantic information and bibliographic information and 3) BERT-GCN (Jeong et al., 2020). Furthermore, to analyze the influence on ranking performance of diverse pretraining corpuses for BERT, we compared our SciBERT reranker with a BERT reranker that was pretrained on a non-science specific corpus (Devlin et al., 2019) and then fine-tuned on the reranking task.

For a fair performance comparison of our reranker with those of other works, we adopted the prefetching strategies from each of these works. On ACL-200 and RefSeer, we tested our SciBERT reranker on the test sets provided in Medić and Šnajder (Medić and Snajder, 2020). For each query in the test set, we prefetched n ($n = 2000$ for ACL-200 and $n = 2048$ for RefSeer) candidates using BM25, and manually added the cited paper as candidate if it was not found by BM25. In other words, we constructed our test set using an "oracle-BM25" with R@$n = 1$. On the FullTextPeerRead dataset, we used our SciBERT reranker to rank all papers in the database without prefetching, in line with the setting in BERT-GCN (Jeong et al., 2020). On our newly proposed arXiv dataset, we fetched the top 2000 candidates for each query in the test set using the "oracle-BM25" as introduced above.

Dataset	Model	avg. prefetch time (ms)	MRR	R@10	R@100	R@200	R@500	R@1000	R@2000
ACL-200	BM25	9.9 ± 20.1	0.138	0.263	0.520	0.604	0.712	0.791	0.859
	Sent2vec	1.8 ± 19.5	0.066	0.127	0.323	0.407	0.533	0.640	0.742
	NNSelect	1.8 ± 3.8	0.076	0.150	0.402	0.498	0.631	0.722	0.797
	HAtten	2.7 ± 3.8	**0.148***	**0.281***	**0.603***	**0.700***	**0.803***	**0.870***	**0.924***
FullText-PeerRead	BM25	5.1 ± 18.6	**0.185***	**0.328***	0.609	0.694	0.802	0.877	0.950
	Sent2vec	1.7 ± 19.6	0.121	0.215	0.462	0.561	0.694	0.794	0.898
	NNSelect	1.7 ± 4.8	0.130	0.255	0.572	0.672	0.790	0.869	0.941
	HAtten	2.6 ± 4.9	0.167	0.306	**0.649***	**0.750***	**0.870***	**0.931***	**0.976***
RefSeer	BM25	216.2 ± 84.9	0.099	0.189	0.398	0.468	0.561	0.631	0.697
	Sent2vec	6.0 ± 20.9	0.061	0.111	0.249	0.306	0.389	0.458	0.529
	NNSelect	4.3 ± 5.5	0.044	0.080	0.197	0.250	0.331	0.403	0.483
	HAtten	6.2 ± 7.3	**0.115***	**0.214***	**0.492***	**0.589***	**0.714***	**0.795***	**0.864***
arXiv	BM25	702.2 ± 104.7	0.118	0.222	0.451	0.529	0.629	0.700	0.763
	Sent2vec	11.3 ± 13.6	0.072	0.131	0.287	0.347	0.435	0.501	0.571
	NNSelect	6.9 ± 4.6	0.042	0.079	0.207	0.266	0.359	0.437	0.520
	HAtten	8.0 ± 4.5	**0.124***	**0.241***	**0.527***	**0.619***	**0.734***	**0.809***	**0.871***

Table 2.2: Prefetching performance. For Tables 2.2-2.4, the asterisks "*" indicate statistical significance ($p < 0.05$) in comparison with the closest baseline in a t-test. The red color indicates a large (> 0.8) Cohen's d effect size (Cohen, 2013).

2.6 Results and Discussion

In this section, we first present the evaluation results of our prefetching and reranking models separately and compare them with baselines. Then, we evaluate the performance of the entire prefetching-reranking pipeline, and analyze the influence of the number of prefetched candidates to be reranked on the overall recommendation performance.

2.6.1 Prefetching Results

Our HAtten model significantly outperformed all baselines (including the strong baseline BM25, Table 2.2) on the ACL-200, RefSeer and the arXiv datasets, evaluated in terms of MRR and R@K. We observed that, first, for larger K, such as $K = 200, 500, 1000, 2000$, the improvement of R@K with respect to the baselines is more pronounced on all four datasets, where the increase is usually larger than 0.1, which means that the theoretical upper bound of the final reranking recall will be higher when using our HAtten prefetching system. Second, the improvements of R@K on large datasets such as RefSeer and arXiv are more prominent than on small datasets such as ACL-200

Model	ACL-200		FullTextPeerRead		RefSeer		arXiv	
	MRR	R@10	MRR	R@10	MRR	R@10	MRR	R@10
NCN	-	-	-	-	0.267	0.291	-	-
DualCon	0.335	0.647	-	-	0.206	0.406	-	-
DualEnh	0.366	0.703	-	-	0.280	0.534	-	-
BERT-GCN	-	-	0.418	0.529	-	-	-	-
BERT-Reranker	0.482	0.736	0.458	0.706	0.309	0.535	0.226	0.399
SciBERT-Reranker	**0.531***	**0.779***	**0.536***	**0.773***	**0.380***	**0.623***	**0.278***	**0.475***

Table 2.3: Comparison of reranking performance on four datasets.

and FullTextPeerRead, which fits well with the stronger need of a prefetching-reranking pipeline on large datasets due to the speed-accuracy tradeoff.

The advantage of our HAtten model is also reflected in the average prefetching time. As shown in Table 2.2, the HAtten model shows faster prefetching than BM25 on large datasets such as RefSeer and arXiv. This is because for HAtten, both text encoding and embedding-based nearest neighbor search can be accelerated by GPU computing, while BM25[2] benefits little from GPU acceleration because it is not vector-based. Although other embedding-based baselines such as Sent2vec and NNSelect also exhibit fast prefetching, our HAtten prefetcher has advantages in terms of both speed and accuracy.

2.6.2 Reranking Results

As shown in Table 2.3, the SciBERT reranker significantly outperformed previous state-of-the-art models on the ACL-200, the RefSeer, and the FullTextPeerRead datasets. We ascribe this improvement to BERT's ability of capturing the semantic relevance between the query text and the candidate text, which is inherited from the "next sentence prediction" pretraining task that aims to predict if two sentences are consecutive. The SciBERT reranker also performed significantly better than its BERT counterpart, suggesting that large language models pretrained on scientific papers' corpus are advantageous for citation reranking.

[2]We implemented the Okapi BM25 (Manning et al., 2008), with $k = 1.2, b = 0.75$.

Dataset	Recommendation Pipeline		Number of reranked candidates				
	Prefetch	Rerank	100	200	500	1000	2000
ACL-200	BM25	$\text{SciBERT}_{\text{BM25}}$	0.457	0.501	0.549	0.577	0.595
	HAtten	$\text{SciBERT}_{\text{HAtten}}$	**0.513***	**0.560***	**0.599***	**0.619***	**0.633***
FullText-PeerRead	BM25	$\text{SciBERT}_{\text{BM25}}$	0.527	0.578	0.639	0.680	0.720
	HAtten	$\text{SciBERT}_{\text{HAtten}}$	**0.586***	**0.651***	**0.713***	**0.739***	**0.757***
RefSeer	BM25	$\text{SciBERT}_{\text{BM25}}$	0.305	0.332	0.365	0.380	0.383
	HAtten	$\text{SciBERT}_{\text{HAtten}}$	**0.362***	**0.397***	**0.428***	**0.443***	**0.454***
arXiv	BM25	$\text{SciBERT}_{\text{BM25}}$	0.333	0.357	0.377	0.389	0.391
	HAtten	$\text{SciBERT}_{\text{HAtten}}$	**0.374***	**0.397***	**0.425***	**0.435***	**0.439***

Table 2.4: The performance of the entire prefetching-reranking pipeline, measured in terms of R@10 of the final reranked document list. We varied the number of prefetched candidates for reranking. For the RefSeer and arXiv datasets, we evaluated performance on a subset of 10K examples from the test set due to computational resource limitations.

2.6.3 Performance of entire Recommendation Pipeline

The evaluation in Section 2.6.2 only reflects the reranking performance because the prefetched candidates are obtained by an oracle-BM25 that guarantees inclusion of the cited paper among the prefetched candidates, even though such an oracle prefetching model does not exist in reality. Evaluating recommendation systems in this context risks overestimating the performance of the reranking part and underestimating the importance of the prefetching step. To better understand the recommendation performance in real-world scenarios, we compared two pipelines: 1) BM25 prefetching + SciBERT reranker fine-tuned on BM25-prefetched candidates, denoted as $\text{SciBERT}_{\text{BM25}}$; 2) HAtten prefetching + $\text{SciBERT}_{\text{HAtten}}$ reranker fine-tuned on HAtten-prefetched candidates. We evaluated recommendation performance by R@10 of the final reranked document list and monitored the dependence of R@10 on the number of prefetched candidates for reranking.

As shown in Table 2.4, the HAtten-based pipeline achieves competitive performance, even when compared with the oracle prefetching model in Section 2.6.2. In particular, on the FullTextPeerRead dataset, using our HAtten-based pipeline, we only need to rerank 100 prefetched candidates to outperform the BERT-GCN model (Table 2.3) that reranked all 4.8k papers in the database.

Compared to the BM25-based pipeline, our HAtten-based pipeline achieves

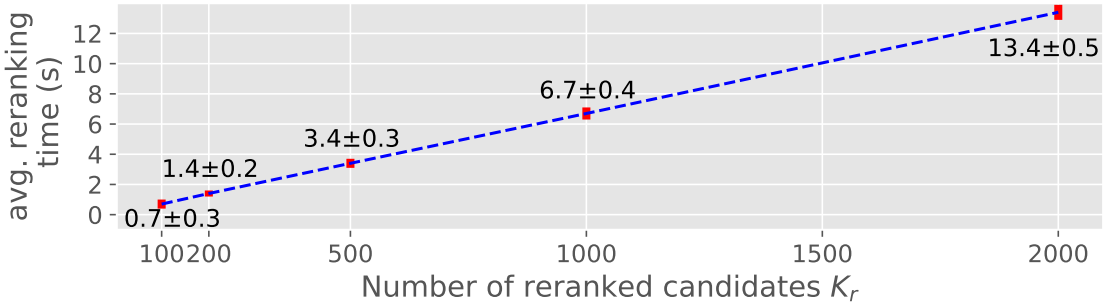


Figure 2.4: The reranking time of the SciBERT reranker linearly increases with the number of reranked candidates K_r, tested on arXiv. In comparison, the prefetching time is invariant of K_r, as the prefetcher always scores and ranks all documents in the database to fetch the candidates to be reranked.

significantly higher R@10 for any given number of prefetched candidates. Our reranker needs to rerank only 200 to 500 candidates to match the recall score of the BM25-based pipeline needing to rerank 2000 candidates. For large datasets like RefSeer and arXiv, such improvements are even more pronounced.

Our pipeline achieves a much higher throughput. For example, on the arXiv dataset, in order to achieve an overall R@10=0.39, the BM25-based pipeline takes 0.7 s (Table 2.2) to prefetch 2000 candidates and it takes another 13.4 s (Figure 2.4) to rerank them, which in total amounts to 14.1 s. In contrast, the HAtten-based pipeline only takes 8 ms to prefetch 200 candidates and 1.4 s to rerank them, which amounts to 1.4 s. This results in a 90% reduction of overall recommendation time achieved by our pipeline.

These findings provide clear evidence that a better-performing prefetching model is critical to a large-scale citation recommendation pipeline, as it allows the reranking model to rerank fewer candidates while maintaining recommendation performance, resulting in a better speed-accuracy tradeoff.

2.7 Conclusion

The speed-accuracy tradeoff is crucial for evaluating recommendation systems in real-world settings. While reranking models have attracted increasing attention for their ability to improve recall and MRR scores, in this paper we show

that it is equally important to design an efficient and accurate prefetching system. In this regard, we propose the HAtten-SciBERT recommendation pipeline, in which our HAtten model effectively prefetches a list of candidates with significantly higher recall than the baseline, which allows our fine-tuned SciBERT-based reranker to operate on fewer candidates with better speed-accuracy tradeoff. Furthermore, by releasing our large-scale arXiv-based dataset, we provide a new testbed for research on local citation recommendation in real-world scenarios.

Chapter 3

MemSum: Extractive Summarization of Long Documents Using Multi-Step Episodic Markov Decision Processes

Originally published as: Gu, Nianlong, Elliott Ash, and Richard Hahnloser. "MemSum: Extractive Summarization of Long Documents Using Multi-Step Episodic Markov Decision Processes." Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2022.

Abstract

We introduce MemSum (Multi-step Episodic Markov decision process extractive SUMmarizer), a reinforcement-learning-based extractive summarizer enriched at each step with information on the current extraction history. When MemSum iteratively selects sentences into the summary, it considers a broad information set that would intuitively also be used by humans in this task: 1) the text content of the sentence, 2) the global text context of the rest of the document, and 3) the extraction history consisting of the set of sentences that

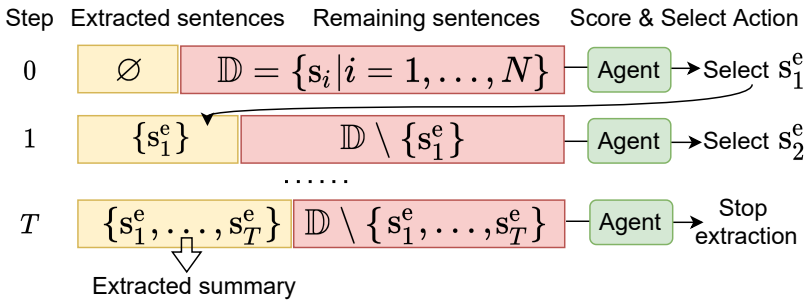


Figure 3.1: We modeled extractive summarization as a multi-step iterative process of scoring and selecting sentences. s_i represents the i_{th} sentence in the document $\mathbb{D}$.

have already been extracted. With a lightweight architecture, MemSum obtains state-of-the-art test-set performance (ROUGE) in summarizing long documents taken from PubMed, arXiv, and GovReport. Ablation studies demonstrate the importance of local, global, and history information. A human evaluation confirms the high quality and low redundancy of the generated summaries, stemming from MemSum's awareness of extraction history.

keywords: Long Document Extractive Summarization; Reinforcement Learning

3.1 Introduction

Automatic text summarization is the task of automatically summarizing a long document into a relatively short text while preserving most of the information (Tas and Kiyani, 2007). Text summarization methods can be categorized into abstractive and extractive summarization (Gambhir and Gupta, 2017; Nenkova and McKeown, 2012). Given a document d consisting of an ordered list of N sentences, *extractive summarization* aims to pick up M ($M \ll N$) sentences as the summary of the document. The extracted summaries tend to be both grammatically and semantically more reliable than abstractive summaries (Liu* et al., 2018; Liu and Lapata, 2019a; Luo et al., 2019; Liao et al., 2020), as they are directly selected from the source text.

Extractive summarization is usually modeled as two sequential phases (Zhou et al., 2018): 1) *sentence scoring* and 2) *sentence selection*. In the sentence scoring

phase, an affinity score is computed for each sentence by neural networks such as bidirectional RNNs (Dong et al., 2018; Narayan et al., 2018; Luo et al., 2019; Xiao and Carenini, 2019) or BERT (Zhang et al., 2019; Liu and Lapata, 2019b). In the sentence selection phase, sentences are selected by either i) predicting a label (1 or 0) for each sentence based on its score, and selecting sentences with label 1 (Zhang et al., 2019; Liu and Lapata, 2019b; Xiao and Carenini, 2019), or ii) ranking sentences based on their scores and selecting the top K sentences as the summary (Narayan et al., 2018), or iii) sequentially sampling sentences without replacement, where the normalized scores of the remaining sentences are used as sampling likelihoods (Dong et al., 2018; Luo et al., 2019).

In these approaches, sentence scores are generally not updated based on the current partial summary of previously selected sentences, indicating a lack of knowledge of *extraction history*. We deem extractive summarizers that are not aware of the extraction history to be susceptible to redundancy in a document, because they will repeatedly add sentences with high scores to a summary, regardless of whether similar sentences have been selected before. And, redundancy leads to performance decreases evaluated by ROUGE F1.

In this paper, we propose to model extractive summarization as a multi-step episodic Markov Decision Process (MDP). As shown in Figure 3.1, at each time step in an episode, we define a *sentence state* composed of three sub-states: 1) the local content of the sentence, 2) the global context of the sentence within the document, and 3) information on the extraction history, including the previously selected set of unordered sentences and the remaining sentences. At each time step, the policy network (agent) takes the current sentence state as input and produces scores used to select an action of either stopping the extraction process or selecting one of the remaining sentences into the candidate summary. Unlike one-step episodic MDP-based models (Narayan et al., 2018; Dong et al., 2018; Luo et al., 2019) that encode the state information only once at the beginning of the episode, in our multi-step policy, the agent updates at each time step the extraction history before selecting an action. Such a step-wise state-updating strategy enables the agent to consider the content of the partial summary when selecting a sentence.

To efficiently encode local and global sentence states, we design an extraction agent based on LSTM networks (Hochreiter and Schmidhuber, 1997). To encode the extraction history and to select actions, we use a reduced number of attention layers (Vaswani et al., 2017) of relatively low dimensionality. These choices enable our model to be easily trainable and to summarize long documents such as scientific papers (Cohan et al., 2018; Huang et al., 2021) or reports (Huang et al., 2021).

The **contributions** of our work are as follows:

- We propose to treat extractive summarization as a multi-step episodic MDP that is aware of the extraction history.
- We show that extraction-history awareness allows our model to extract more compact summaries than models without history awareness and behave more robustly to redundancies in documents.
- Our model outperforms both extractive and abstractive summarization models on PubMed, arXiv (Cohan et al., 2018), and GovReport (Huang et al., 2021) datasets.
- Finally, human evaluators rate the MemSum summaries to be of higher quality than those from a competitive approach, especially by virtue of lower redundancy[1].

3.2 Related Work

Extraction history awareness was previously considered in NeuSum (Zhou et al., 2018), where a GRU encoded previously selected sentences into a hidden vector that then was used to update the scores of the remaining sentences to bias the next selection. NeuSum contains no stopping mechanism and therefore it can only extract a fixed number of sentences, which likely is sub-optimal. Also, the potential benefits of extraction history have not been quantified and so the idea remains unexplored to a large extent.

[1]Our code and data are available at https://github.com/nianlonggu/MemSum

Recently, BERT-based extractors such as MatchSum (Zhong et al., 2020) achieved SOTA performance in extractive summarization of relatively short documents from the CNN/DM (Hermann et al., 2015) dataset. However, the quadratic computational and memory complexities (Huang et al., 2021) of such models limit their scalability for summarizing long documents with thousands of tokens, which is common for scientific papers and government reports. Although large pre-trained transformers with efficient attention (Huang et al., 2021) have been adapted for abstractive summarization of long documents, we believe that extractive summarization is more faithful in general, which is why we chose an extractive approach.

3.3 Model

This section outlines the multi-step episodic MDP policy for extractive summarization.

3.3.1 Policy Gradient Methods

In an episodic task with a terminal state (i.e. *end of summary*), policy gradient methods aim to maximize the objective function $J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R_0]$, where the return $R_t = \sum_{k=t+1}^{T} r_k$ is the cumulative reward from time $t+1$ until the end of the episode when the summary is complete. In applications of RL to extractive summarization, the instantaneous reward r_t is zero except at the end of the episode when the final reward r is computed according to Equation (3.1), so $R_t \equiv R_0 = r$. The reward r is usually expressed as (Dong et al., 2018):

$$r = \frac{1}{3}(\text{ROUGE-1}_f + \text{ROUGE-2}_f + \text{ROUGE-L}_f) \tag{3.1}$$

According to the REINFORCE algorithm (Williams, 1992), the policy gradient is defined as:

$$\nabla J(\boldsymbol{\theta}) = \mathbb{E}_{\pi}[R_t \nabla \log \pi(A_t|S_t, \boldsymbol{\theta})], \tag{3.2}$$

where $\pi(A_t|S_t, \boldsymbol{\theta})$ denotes the likelihood that at time step t the policy $\pi_{\boldsymbol{\theta}}$ selects action A_t given the state S_t. With α as the learning rate, the parameter update

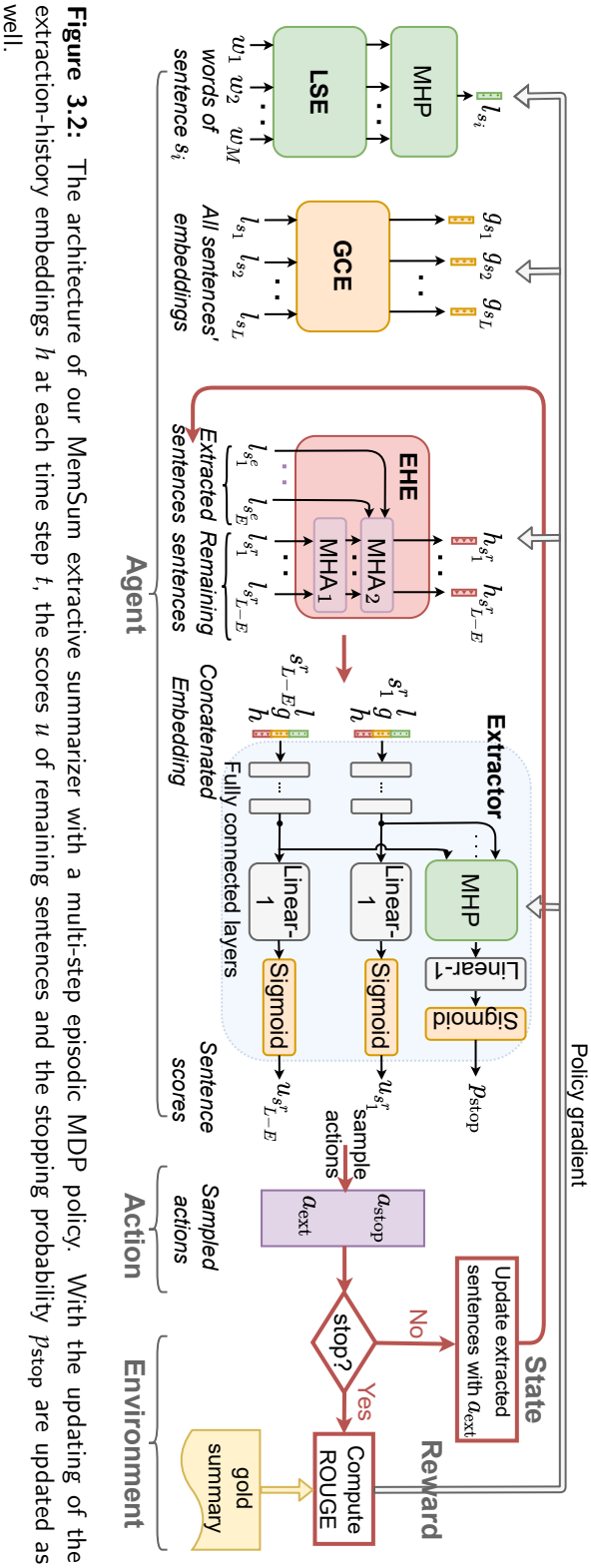


Figure 3.2: The architecture of our MemSum extractive summarizer with a multi-step episodic MDP policy. With the updating of the extraction-history embeddings h at each time step t, the scores u of remaining sentences and the stopping probability p_{stop} are updated as well.

rule is (Sutton and Barto, 2018):

$$\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t + \alpha R_t \nabla \log \pi(A_t|S_t, \boldsymbol{\theta}_t), \tag{3.3}$$

3.3.2 Multi-step Episodic MDP Policy

Different from one-step episodic MDP policies (Narayan et al., 2018; Dong et al., 2018; Luo et al., 2019) that extract the entire summary via a single action, we define an episode, i.e., the generation of a summary, consisting of multiple time steps. At each time step t, corresponding to extracting sentence number t, the action A_t is either to stop extraction or to select a sentence s_{a_t} from the remaining sentences. The agent's policy is:

$$\begin{aligned} \pi(A_t|S_t, \boldsymbol{\theta}_t) &= p(\text{stop}|S_t, \boldsymbol{\theta}_t) p(a_t|\text{stop}, S_t, \boldsymbol{\theta}_t) \\ p(a_t|\text{stop}, S_t, \boldsymbol{\theta}_t) &= \begin{cases} \frac{u_{a_t}(S_t, \boldsymbol{\theta}_t)}{\sum_{j \in I_t} u_j(S_t, \boldsymbol{\theta}_t)} & \text{if stop = false} \\ \frac{1}{|I_t|} & \text{if stop = true,} \end{cases} \end{aligned} \tag{3.4}$$

where I_t denotes the index set of remaining sentences at time step t. If the agent does not stop, it first computes a score u_j for each remaining sentence and samples a sentence s_{a_t} according to the probability distribution of normalized scores. When the agent stops the extraction, no sentence is selected and the conditional likelihood $p(a_t|\text{stop=false}, S_t, \boldsymbol{\theta}_t)$ is set to $\frac{1}{|I_t|}$ (where $|I_t|$ represents the number of remaining sentences at time t), which is independent of the policy parameters to prohibit the gradient from being passed to the policy parameters via the conditional likelihood. After calculating the reward according to Equation (3.1), the policy parameters are updated according to Equation (3.3) (for all time steps).

3.3.3 Policy Network

The state S_t in Equation (3.4) is designed to be informative on: 1) the local content of the sentence, 2) the global context of the sentence within the document, and 3) the current extraction history. To encode these three properties in the state, we use a local sentence encoder, a global context encoder, and

an extraction history encoder, respectively. Subsequently, the state is mapped by an extractor to an output score for each of the remaining sentences and the extraction stop signal. The overall framework of our model is depicted in Figure 3.2.

In the **Local Sentence Encoder** (LSE), ordered words $(w_1, w_2, \dots w_M)$ in a sentence s_i are first mapped onto word embeddings using a word embedding matrix. Subsequently, a N_l-layer bi-directional LSTM (Hochreiter and Schmidhuber, 1997) transforms the word embeddings and maps them onto sentence embeddings l_{s_i} via a multi-head pooling layer (MHP) (Liu and Lapata, 2019a).

The **Global Context Encoder** (GCE) consists of a N_g-layer bi-LSTM that takes the L local sentence embeddings $(l_{s_1}, l_{s_2}, \dots l_{s_L})$ as inputs and produces for each sentence s_i an embedding g_{s_i} that encodes global contextual information such as the sentence's position in the document and information on neighboring sentences.

The **Extraction History Encoder** (EHE) encodes the extraction history information and produces the extraction history embedding $h_{s_i^r}$ for each remaining sentence s_i^r. The EHE is composed of a stack of N_h identical layers. Within one layer, there are two multi-head attention sublayers, as contained in the transformer decoder in Vaswani et al. (2017). One sublayer is used to perform multi-head self-attention (MHA) among the local embeddings of the remaining sentences, so that each remaining sentence can capture the context provided by other remaining sentences. The other attention sublayer is used to perform multi-head attention over the embeddings of extracted sentences to enable each remaining sentence to attend to all the extracted sentences. The output of the two attention sublayers, one for each remaining sentence, captures the contextual information of both extracted and remaining sentences. The final output of the N_h^{th} layer of the EHE constitutes the extraction history embedding, one for each remaining sentence.

There is no positional encoding and the EHE produces the extraction history embeddings non-autoregressively by attending to both precedent and subsequent positions. Consequently, the extraction history embeddings $h_{s_i^r}$ for the

remaining sentences are invariant to the order of the previously selected sentences. We believe that the sequential information of previously selected sentences is not crucial for reducing redundancy and for deciding whether to stop extraction or not.

The **Extractor** computes the score of each remaining sentence and outputs an extraction stop signal. As input to the extractor, we form for each of the remaining sentences s_i^r an aggregated embedding by concatenating the local sentence embedding $l_{s_i^r}$, the global context embedding $g_{s_i^r}$, and the extraction history embedding $h_{s_i^r}$. As shown in Figure 3.2, to produce the score $u_{s_i^r}$, the concatenated embedding of remaining sentence s_i^r is passed to fully connected layers with ReLU activation and then projected to a scalar by a Linear-1 layer followed by a sigmoid function. Note that the same fully connected layers are applied identically to all remaining sentences. We deem that the extractor can learn to stop extraction based on the remaining sentences' states. Therefore, we apply an MHP to the last hidden vectors of all remaining sentences to output a single vector. This vector is then passed to a linear layer with a sigmoid function, producing a stopping probability p_{stop}.

3.3.4 Training

We train the parameterized policy network according to the update rule in Equation (3.3). At each training iteration, an episode is sampled to compute the final return r and the action probabilities $\pi(A_t|S_t, \boldsymbol{\theta}_t)$ for all time steps t. An example episode with T extracted sentences looks like: $(S_0, s_{a_0}, \ldots, S_{T-1}, s_{a_{T-1}}, S_T, A_{\text{stop}}, r)$, where S_t represents the concatenated state information introduced in Section 3.3.3, s_{a_t} represents the selection of sentence a_t, A_{stop} represents the extraction stops at the final time step T, and r is the reward as defined in Equation (3.1). To encourage the agent to select compact summaries, we multiply the final reward r by a length penalty term $1/(T+1)$ (Luo et al., 2019). Consequently, the return $R_t \equiv \frac{r}{T+1}$.

Algorithm 1 summarizes the training procedure of MemSum. We initialize the extraction history embeddings to $\mathbf{0}$, because at $t = 0$ no sentences have been

Algorithm 1 The training algorithm.

Parameters: learning rate α

for each document-summary pair (D_i, G_i) **do**
  LSE outputs local sent. embed $l_{s_1},\ldots,l_{s_L}$
  GCE outputs global context embed $g_{s_1},\ldots,g_{s_L}$
  Sample an episode $S_0,s_{a_0},\ldots,S_{T-1},s_{a_{T-1}}$, S_T,A_{stop},r from the high-ROUGE episodes set $\mathbb{E}_p$ of document D_i
  for each time step: t = 0,1,...,T: **do**
    if $t > 0$ **then**
      EHE outputs extraction history embed $h_{s_1^r},\ldots,h_{s_{L-E_t}^r}$ for remaining sentences
    else
      Initialize $h_{s_1^r},...,h_{s_{L-E_0}^r}$ to $\mathbf{0}$
    Extractor outputs scores $u_{s_1^r},...,u_{s_{L-E_t}^r}$ for remaining sentences and outputs p_{stop}
    Compute the action probability $\pi(A_t|S_t,\boldsymbol{\theta})$ according to Equation (3.4)
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \frac{r}{T+1} \nabla \log \pi(A_t|S_t,\boldsymbol{\theta})$

extracted. E_t represents the number of sentences that have been extracted into the summary up to time step t. Following the strategy in Narayan et al. (2018) and Mohsen et al. (2020), instead of sampling an episode following the current policy $\pi(\cdot|\cdot,\boldsymbol{\theta}_t)$, we sample an episode from a set $\mathbb{E}_p$ of episodes with high ROUGE scores, which enables the agent to quickly learn from optimal policies and to rapidly converge. Details on creating a set of high-ROUGE episodes for training are described in Appendix E.

3.4 Experiments

In this section, we report implementation details of our model and describe the datasets used for training and for evaluation.

Datasets. The documents to be summarized in the PubMed and arXiv datasets (Cohan et al., 2018) are the full bodies of scientific papers and the gold summaries are the corresponding abstracts. Zhong et al. (2020) proposed a trun-

Datasets	avg. doc. length		avg. summ. length		# of doc.-summ. pairs		
	# of words	# of sent.	# of words	# of sent.	Train	Valid	Test
PubMed	2,730	88	181	7	116,937	6,633	6,658
arXiv	5,206	206	238	10	202,880	6,436	6,440
$\text{PubMed}_{\text{trunc}}$	408	13	185	7	83,233	4,676	5,025
GovReport	7,932	307	501	18	17,517	974	973
CNN/DM	692	35	49	4	-	-	-

Table 3.1: An overview of datasets used in this paper. We count only strings composed of letters and numbers for # of words.

cated version of the PubMed dataset ($\text{PubMed}_{\text{trunc}}$ for simplicity) by defining a doument as the introduction section of a paper. The GovReport dataset (Huang et al., 2021) contains U.S. government reports with gold summaries written by experts. Except $\text{PubMed}_{\text{trunc}}$, all the other datasets contain significantly longer documents than the popular dataset CNN/DM (Table 3.1).

Baselines. Extractive baselines include Lead (directly using the first several sentences as the summary) (Gidiotis and Tsoumakas, 2020), SummaRuNNer (Nallapati et al., 2017), Atten-Cont (Xiao and Carenini, 2019), Sent-CLF and Sent-PTR (Pilault et al., 2020), MatchSum (Zhong et al., 2020), and the NeuSum model (Zhou et al., 2018) that we trained on our datasets.

Abstractive summarization models include PEGASUS (Zhang et al., 2020), BigBird (Zaheer et al., 2020), Dancer (Gidiotis and Tsoumakas, 2020), and Hepos (Huang et al., 2021) that achieved the state-of-the-art in long document summarization using a large-scale pretrained BART model (Lewis et al., 2020) with memory-efficient attention encoding schemes including Locality Sensitive Hashing (Kitaev et al., 2020) (Hepos-LSH) and Sinkhorn attention (Hepos-Sinkhorn). We also present the performance of the oracle extraction model based on the greedy approach (Nallapati et al., 2017) which sequentially selects from the document the sentence that maximally improves the average of R-1 and R-2 of selected sentences.

Implementation Details. We computed local sentence embeddings using

pretrained Glove word embeddings (Pennington et al., 2014) of dimension $d = 200$, keeping the word embeddings fixed during training. For the LSE, we used $N_l = 2$ bi-LSTM layers and for the GCE $N_g = 2$. For the EHE, we used $N_h = 3$ attention layers, and we set the number of attention heads to 8 and the dimension of the feed-forward hidden layer to 1024; during training we set the dropout rate to 0.1. The extractor consisted of 2 fully-connected hidden layers with output dimensions $2d$ and d, respectively.

We trained our model using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ (Kingma and Ba, 2015), fixed learning rate $\alpha = 1e^{-4}$, and weight decay $1e^{-6}$. The training was stopped when the validation performance started to degrade. During validating and testing, the agent extracted sentences in a deterministic way: after computing the scores $u_{s_i^r}$ for the remaining sentences and the stop likelihood p_{stop}, the agent stopped the extraction if $p_{\text{stop}} \geq p_{\text{thres}}$ or if the maximum admissible number N_{max} of extracted sentences was reached; otherwise, the agent selected the sentence with the largest score. The model was trained on eight RTX 2080 Ti GPUs.

On the validating datasets we selected the best checkpoint of each model and determined the optimal N_{max} and stopping criterion p^*_{thres}. For Pubmed, arXiv, $\text{Pubmed}_{\text{trunc}}$, and GovReport, N_{max} was set to 7, 5, 7, and 22, and p^*_{thres} was set to 0.6, 0.5, 0.8, and 0.6, respectively. For the detailed selection procedure of the optimal stopping threshold, see Appendix D. Information on reproducibility is available in Appendix I.

Evaluation. We evaluated the performance of our model using F_1 ROUGE (Lin, 2004), including ROUGE-1,2, and L for measuring unigram, bigram, and longest common subsequence. We also conducted human evaluation in Section 3.5.4.

3.5 Results and Discussion

Here we present the results on various extractive summarization tasks and analyze the contribution of different modules via ablation studies.

Model	PubMed			arXiv		
	R-1	R-2	R-L	R-1	R-2	R-L
ORACLE	61.99	34.95	56.76	60.00	30.60	53.03
Extractive summarization baselines						
Lead-10	37.45	14.19	34.07	35.52	10.33	31.44
SummaRuNNer	43.89	18.78	30.36	42.81	16.52	28.23
Atten-Cont	44.85	19.70	31.43	43.62	17.36	29.14
Sent-CLF	45.01	19.91	41.16	34.01	8.71	30.41
Sent-PTR	43.30	17.92	39.47	42.32	15.63	38.06
NeuSum	47.46	21.92	42.87	47.49	21.56	41.58
Abstractive summarization baselines						
PEGASUS	45.97	20.15	41.34	44.21	16.95	38.83
BigBird	46.32	20.65	42.33	46.63	19.02	41.77
Dancer	46.34	19.97	42.42	45.01	17.60	40.56
Hepos-Sinkhorn	47.93	20.74	42.58	47.87	20.00	41.50
Hepos-LSH	48.12	21.06	42.72	48.24	20.26	41.78
MemSum (ours)	**49.25***	**22.94***	**44.42***	**48.42**	**20.30**	**42.54***

Table 3.2: Results on the PubMed and arXiv test sets. "*" indicates that they are statistically significant in comparison to the closest baseline with a 95% bootstrap confidence interval estimated by the ROUGE script[2].

3.5.1 Results Comparison

By comparing with extractive baselines on the PubMed and arXiv datasets, we observed that models utilizing extraction history, such as NeuSum and our MemSum, perform significantly better than other models, revealing the effectiveness of the extraction history. MemSum also significantly outperformed NeuSum, suggesting a *better utilization of extraction history*, which we ascribed to the following factors: 1) In MemSum, we treat stopping extraction also as an action and train the policy network to output a stopping probability. Therefore, MemSum is able to automatically stop extracting at an optimal time step based on extraction history, while NeuSum can only extract a predefined number of sentences; 2) With the policy gradient method REINFORCE we can train MemSum to maximize the ROUGE score directly, while in NeuSum the loss was set to the KL-divergence between the model-computed sentence scores and the ROUGE score gains at each step, which is less intuitive. We further

Model	PubMed$_{trunc}$			GovReport		
	R-1	R-2	R-L	R-1	R-2	R-L
ORACLE	45.12	20.33	40.19	75.56	45.91	72.51
Extractive summarization baselines						
Lead	37.58	12.22	33.44	50.94	19.53	48.45
MatchSum	41.21	14.91	36.75	-	-	-
NeuSum	-	-	-	58.94	25.38	55.80
Abstractive summarization baselines						
Hepos-LSH	-	-	-	55.00	21.13	51.67
Hepos-Sinkhorn	-	-	-	56.86	22.62	53.82
MemSum (ours)	**43.08***	**16.71***	**38.30***	**59.43***	**28.60***	**56.69***

Table 3.3: Results on PubMed$_{trunc}$ and GovReport.

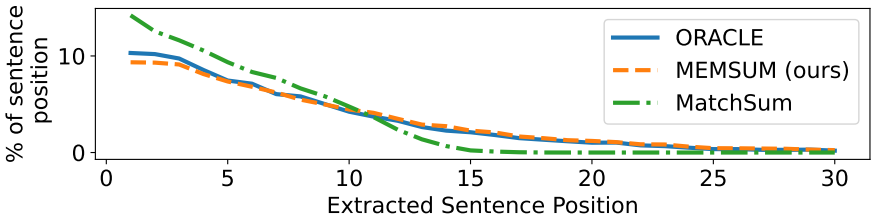


Figure 3.3: The position distribution of extracted sentences in the PubMed$_{trunc}$ dataset.

compare MemSum with NeuSum via human evaluation in Section 3.5.4.

We observed that the ROUGE performance on the PubMed$_{trunc}$ dataset is significantly lower than that on the PubMed dataset, with a 16.87 drop in R-1 for the extractive oracle and a 6.23 drop in R-1 for MemSum, *indicating that the introduction section is not sufficient to generate summaries close to the ground truth (abstracts).* Even so, our model still significantly outperformed MatchSum on PubMed$_{trunc}$, and we attribute this improvement to the fact that MatchSum truncates the introduction section further to 512 tokens because it needs to compute document embeddings using Bert. Consequently, MatchSum extracts sentences mainly from the first 15 sentences of the document, while our MemSum produces a similar distribution of extracted sentence positions as the extractive oracle, Figure 3.3. Thus, *summarizing long documents is a non-trivial task*, and models that work well on summarizing short documents (e.g., CNN/DM)

[2] https://pypi.org/project/rouge-score/

Human-written Summary:
(...) While CMS is generally required to disallow, or ***recoup, federal funds*** from states for ***eligibility-related improper payments*** if the state's ***eligibility error rate exceeds 3 percent***, it has not done so for decades, (...) CMS ***issued revised procedures through which it can recoup funds for eligibility errors, beginning in fiscal year 2022***. (...)
Hepos-Sinkhorn (abstractive):
(...) The selected states also reported that they did not have adequate processes to address these issues. CMS has taken steps to improve its oversight of the Medicaid program, including issuing guidance to states on the use of MAGI-exempt bases for determining eligibility, but these efforts have not been fully implemented. (...)
MemSum (ours, extractive):
(...) In 1983, CMS implemented its statutory requirement to ***recoup funds*** associated with Medicaid ***eligibility-related improper payments*** for states with an ***eligibility error rate above 3 percent*** through its MEQC program. (...) However, the agency has ***introduced new procedures through which it can, under certain circumstances, begin to recoup funds based on eligibility errors in fiscal year 2022***. (...)

Table 3.4: Comparison of the summary extracted by MemSum and the summary abstractively generated by Hepos-Sinkhorn (Huang et al., 2021). Compared with the abstractive summary, the MemSum-extracted summary has higher overlap with the human-written summary.

may fail to generalize to long documents.

MemSum also significantly outperformed the state-of-the-art abstractive summarization model Hepos as measured by ROUGE scores, especially on the GovReport dataset. A comparison of an exemplary MemSum-extracted summary and the corresponding Hepos-Sinkhorn-generated summary from the GovReport dataset (Table 3.4) is consistent with the ROUGE comparison, showing that the MemSum-extracted summary is more accurate than the Hepos-Sinkhorn-generated summary and has higher overlap with the gold summary. We deem that this particularly good extraction performance on the GovReport dataset results from the higher "extractiveness" of the gold summaries in the GovReport dataset compared to other datasets, which may be due in part to technical language being difficult to abstractively summarize without a change in meaning. This is evidenced by the fact that the ROUGE scores of the extractive oracle on the GovReport dataset (Table 3.3) are higher than those of the PubMed and arXiv datasets (Table 3.2). Therefore, *extractive summarization may be more proper than abstractive summarization due to the requirement of stringent faithfulness of government report summaries*.

Model	R-1	R-2	R-L
MemSum	**49.25**	**22.94**	**44.42**
MemSum w/o LSE	48.12	22.04	43.36
MemSum w/o GCE	46.85	20.31	41.95
MemSum w/o EHE	48.08	22.77	43.55
MemSum with GRU-EHE	49.11	22.86	44.28
MemSum w/o auto-stop	48.25	22.63	43.70
MemSum with "STOP"	47.18	21.81	42.20

Table 3.5: Ablation study on the PubMed dataset.

3.5.2 Ablation Test

We conduct ablation studies by comparing the full MemSum model with the following *variations in structures*: 1) MemSum w/o LSE, where we obtain local sentence embeddings by replacing the bi-LSTM based LSE by simple averages of word embeddings; 2) MemSum w/o GCE where we remove the GCE; 3) MemSum w/o EHE where we remove EHE, compute the scores for all sentences in one step, and samples sentences following the BanditSum policy (Dong et al., 2018); 4) MemSum with GRU-EHE where we use a GRU to encode previously extracted sentences at each time step, and uses the last hidden state as the extraction history embedding for all remaining sentences, following Zhou et al. (2018).

Meanwhile, we also tested two variations that adopted *different stopping mechanisms*: 1) MemSum w/o auto-stop that does not stop extraction automatically based on p_{stop}, but that extracts a fixed number of sentences; 2) MemSum with "STOP" that inserts a special stop sentence (e.g. "STOP") into the document, and stops extraction once the agent selects this sentence.

Contribution of Modules. Removing GCE has a greater impact on performance than removing LSE (Table 3.5), suggesting that modeling global contextual information is more critical than modeling local sentence information in our MemSum framework, which contrasts with the result that modeling local sentence information is more important in the Atten-Cont (Xiao and Carenini, 2019) framework. Furthermore, we observed a significant performance degradation when removing EHE, but no significant difference between MemSum

Model	R-1	R-2	R-L	duplicate percentage
MemSum	**49.16**	**22.78**	**44.39**	0%
MemSum w/o auto-stop	48.21	22.59	43.76	0%
MemSum w/o EHE	42.82	18.18	36.68	41%
MemSum w/o EHE +3gram blocking	46.85	19.93	42.40	0%

Table 3.6: Performance on the redundant PubMed dataset.

and MemSum with GRU-EHE, indicating that EHE is necessary, but our *MemSum policy is not strongly dependent on the specific structure of this module* (e.g., attention-based or RNN-based).

Influence of Stopping Mechanisms. MemSum w/o auto-stop achieves lower ROUGE scores than MemSum, revealing the necessity of auto stopping in our MemSum architecture. Meanwhile, MemSum with "STOP" produced summaries with fewer extracted sentences (3.9 vs. 6.0 sentences on average) and significantly lower ROUGE scores. We attribute this reduction to the predictable positive reward obtained from selecting the special stop sentence that ends an episode, which leads to a preference for this final action and increases the likelihood of taking this action prematurely.

3.5.3 History Awareness Avoids Redundancy

We hypothesized that the extraction history allows MemSum to avoid sentences that are similar to existing sentences in the current partial summary, intuitively mimicking what humans do when extractively summarizing documents. To verify this, we created a redundant PubMed dataset in which we repeated each sentence in the document, with the replicated sentences immediately following the originals. On this dataset, we trained and tested MemSum and MemSum w/o EHE (no history awareness), and we compared different models in terms of ROUGE scores and average *duplicate percentage* that is defined as the average percentage of the duplicated sentences among all extracted sentences in a summary.

As reported in Table 3.6, for MemSum w/o EHE, on average 41% of sentences in the extracted summaries were duplicated. Along with the high duplicate ratio came a significant decrease in ROUGE score. By contrast, the performance of the full MemSum model with history awareness was only slighted affected when comparing the results of the MemSum on the PubMed dataset (Table 3.2) and on the redundant PubMed dataset (Table 3.6).

Meanwhile, using the Trigram Blocking method that skips a sentence if it has a trigram that overlaps with the current summary (Liu and Lapata, 2019b) is also successful in avoiding repetitive sentences. However, the ROUGE scores associated with Trigram Blocking were significantly lower than those of the MemSum with awareness of extraction history. In summary, the history-aware MemSum model spontaneously learns an optimized strategy to avoid redundant sentences without explicit human guidance or crude rules, and thus shows better performance.

Case Study: How does MemSum Avoid Redundancy?

We let MemSum summarize a document sampled from the test set of the redundant PubMed dataset and monitored the sentence scores produced by the Extractor during each extraction step. The results are shown in Figure 3.4. At time step 0, the 10_{th} sentence obtained the maximum score and was thus selected into the summary. At time step 1, we noticed that the 11_{th} sentence, which is a replica of the 10_{th} sentence, had a score close to zero. The same was also true for the other selected sentences and their following sentences, revealing competent repetition avoidance of the Extractor. Because the EHE is insensitive to the extraction order and to sentence position information, as described in Section 3.3.3, we can conclude that the full MemSum avoids redundancy by evaluating the similarity between selected and remaining sentences, rather than by "remembering" selected sentences' positions.

3.5.4 Human Evaluation

We conducted human evaluation following Wu and Hu (2018); Dong et al. (2018); Luo et al. (2019). For each document sampled from the test set of the

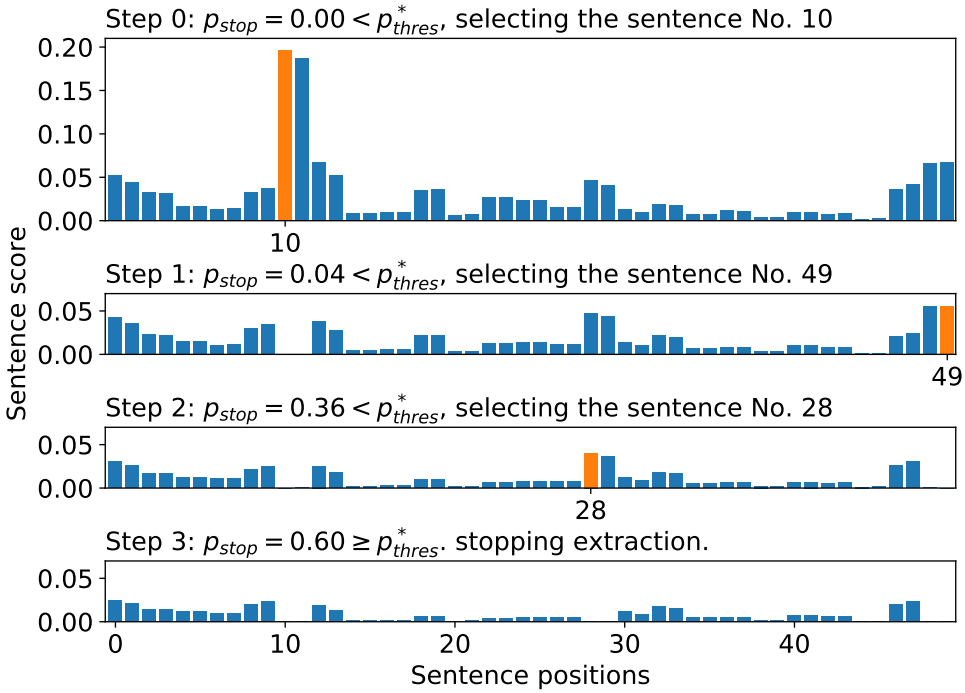


Figure 3.4: The sentence scores of 50 sentences computed by MemSum at extraction steps 0 to 3. In the document, there is artificial redundancy in that the $(2n)_{\text{th}}$ and the $(2n+1)_{\text{th}}$ sentences are identical $(n = 0, 1, ..., 24)$.

PubMed dataset, we provide a reference summary, and volunteers are asked to rank a pair of randomly ordered summaries produced by two models according to three criteria: non-redundancy, coverage, and overall quality. The better model will be ranked #1 while the other is ranked #2, and if both models extract the same summary, then they will both get the #1 rank. In experiment 1, we compared NeuSum, which always extracts 7 sentences, and MemSum, which extracts a flexible number of sentences thanks to automatic stopping. In experiment 2, we discounted for differences in the number of extracted sentences by making MemSum w/o auto-stop to also extract 7 sentences. A user-friendly interactive web interface was implemented to assist the evaluation process, with details in Appendix G.

Table 3.7 reports the human evaluation results for both experiments. Both MemSum and MemSum w/o auto-stop ranked significantly higher (p¡0.005)

Criteria	Experiment I		Experiment II	
	NeuSum	MemSum	NeuSum	MemSum w/o auto-stop
overall	1.58	**1.37**	1.57	**1.38**
coverage	**1.46**	1.49	**1.44**	1.51
non-redundancy	1.67	**1.28***	1.65	**1.30***
avg. summ. length				
# of sentences	7.0	**5.6***	7.0	7.0
# of words	248.8	**189.3***	263.6	**239.5***

Table 3.7: The average ranking of NeuSum and MemSum is reported. The smaller the ranking, the better the model. Four volunteers participated in these experiments, and evaluated 67 and 63 pairs of summaries in Experiment 1 and 2, respectively. "*" indicates statistical significance (p¡0.005) in a Wilcoxon signed-rank test (Woolson, 2008).

than NeuSum in terms of non-redundancy and achieved a better average overall quality. In terms of word count, MemSum produces shorter summaries than NeuSum in both experiments, even though both models extract the same number of sentences in experiment 2. These results show that redundancy avoidance of MemSum is particularly good, even without the auto-stop mechanism. The slightly better performance of NeuSum in terms of coverage needs to be weighed against it extracting significantly longer summaries. Note that neither NeuSum nor our model is trained to optimize the order of the extracted sentences. Therefore, we did not use fluency, which depends on sentence order, as a metric for human evaluation. Improving the fluency of the extracted summaries will be the subject of our future research.

3.6 Conclusion

Extractive summarization can be achieved effectively with a multi-step episodic Markov decision process with history awareness. Using encoders of local sentence, global context, and extraction history, MemSum is given information that is intuitively also used by humans when they summarize a document. Awareness of the extraction history helps MemSum to produce compact summaries and to be robust against redundancy in the document. As a lightweight model (Appendix C), MemSum outperforms both extractive and abstractive

baselines on diverse long document summarization tasks. Because MemSum achieves SOTA performance on these tasks, MDP approaches will be promising design choices for further research.

3.7 Appendices

A Computing Hardware

We trained our MemSum model and its variations on 8 NVIDIA GeForce RTX 2080 Ti 11GB GPUs. During testing, we used a single NVIDIA TITAN X Pascal 12GB GPU.

B Comparison of Validating and Testing Performance

We compare the validating and testing performance of the MemSum model on the following datasets: PubMed (Cohan et al., 2018), arXiv (Cohan et al., 2018), and GovReport (Huang et al., 2021). The results are reported in Table 3.8.

C Summarization Time

We analyzed the average time taken by MemSum to extractively summarize a source document from the test set. The average summarizaion time is positively correlated with the document length and the number of extracted sentences, Table 3.9. On the one hand, on longer documents, it takes longer to compute the scores of remaining sentences, which delays the action of either stopping extraction or selecting a sentence. On the other hand, the more sentences must be extracted, the more actions are needed of selecting sentences within an episode.

D Selection of optimal stopping threshold

The stopping threshold p_{thres} is an important hyperparameter that sets the stopping probability in an episode, as described in the Implementation Details. We determined the optimal stopping threshold p^*_{thres} as follows: For

Datasets	Validating			Test		
	R-1	R-2	R-L	R-1	R-2	R-L
PubMed	49.14	22.92	44.33	49.25	22.94	44.42
arXiv	48.23	20.17	42.31	48.42	20.30	42.54
$\text{PubMed}_{\text{trunc}}$	43.46	16.77	38.65	43.08	16.71	38.30
GovReport	59.29	28.57	56.46	59.43	28.60	56.69

Table 3.8: Validating and testing scores of the MemSum model tested on the PubMed, the arXiv and the GovReport datasets.

Datasets	avg. doc. length (words)	Avg. extractive summ. length (# sentences)	Avg. extractive summ. time (ms)
PubMed	2,730	6.0 ± 1.2	91.7 ± 8.6
arXiv	5,206	4.8 ± 0.5	114.0 ± 5.0
$\text{PubMed}_{\text{trunc}}$	408	5.3 ± 1.4	27.7 ± 4.6
GovReport	7,932	21.7 ± 1.8	197.0 ± 14.8

Table 3.9: Average extractive summarization time of MemSum on different datasets.

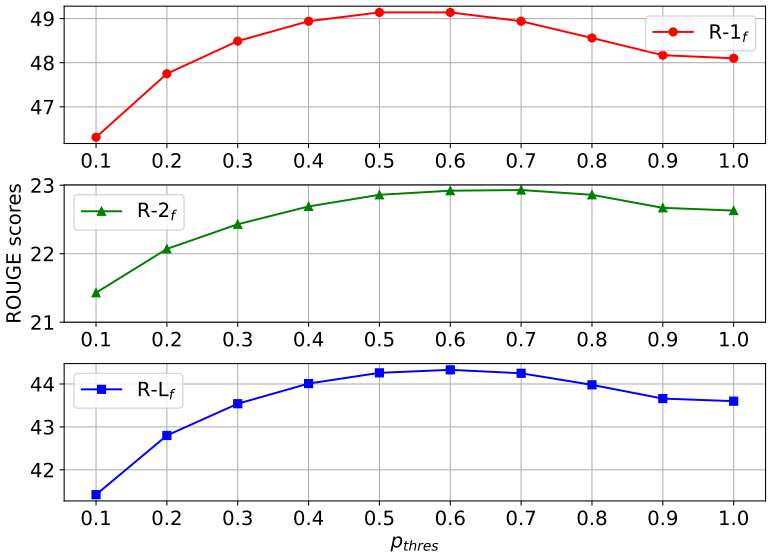


Figure 3.5: The ROUGE scores for different stopping thresholds p_{thres} on the PubMed validating set.

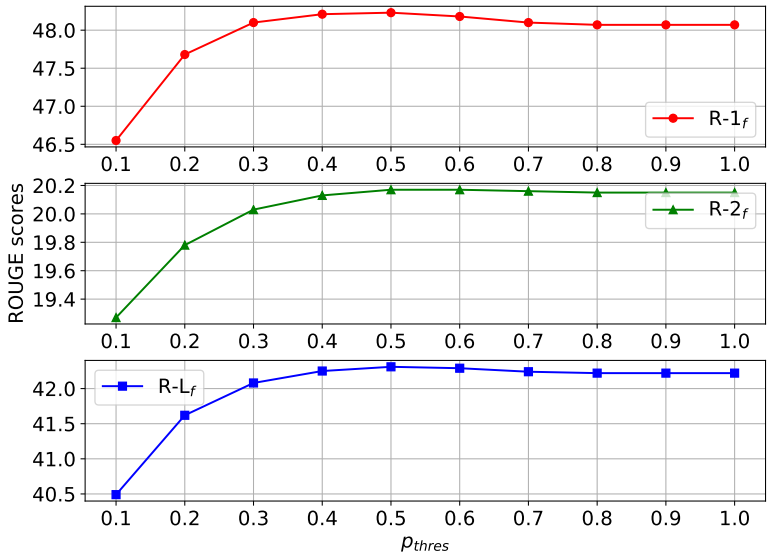


Figure 3.6: The ROUGE scores for different stopping thresholds p_{thres} on the arXiv validating set.

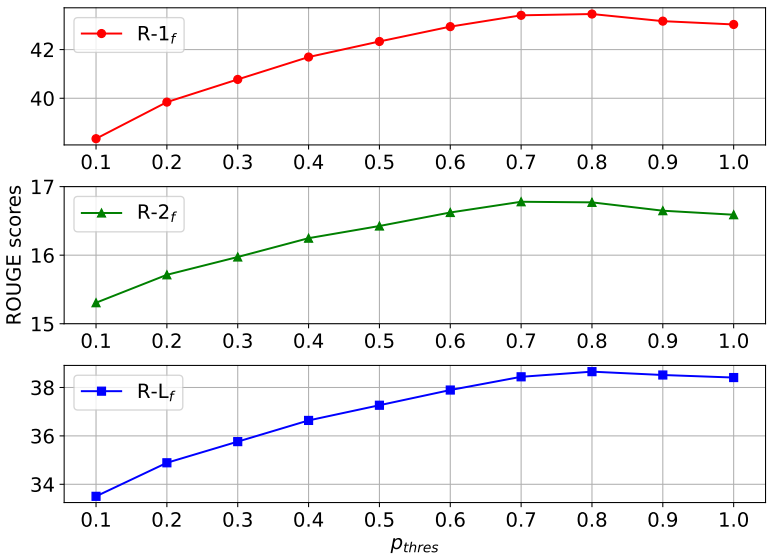


Figure 3.7: The ROUGE scores for different stopping thresholds p_{thres} on the PubMed$_{\text{trunc}}$ validating set.

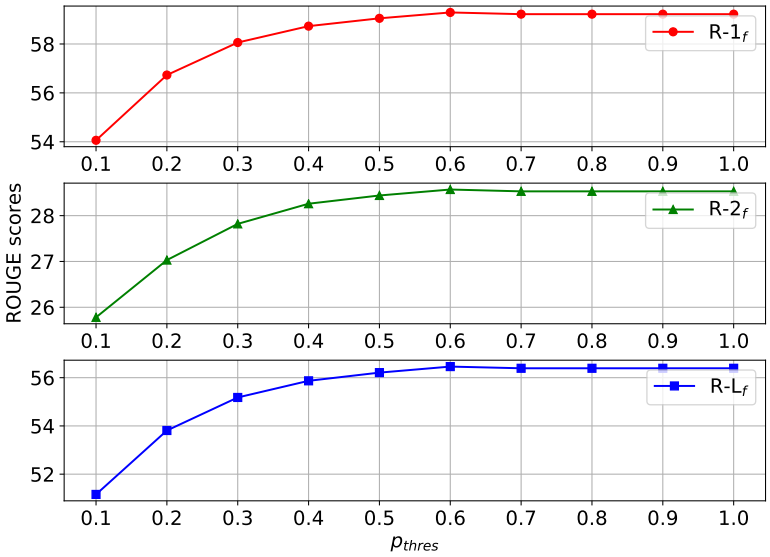


Figure 3.8: The ROUGE scores for different stopping thresholds p_{thres} on the GovReport validating set.

each data set and each stopping threshold $p_{\text{thres}} \in \{0.1, 0.2, \ldots, 1.0\}$, we chose as optimal stopping threshold p^*_{thres} the one with maximal ROUGE score on the corresponding validating set.

The ROUGE scores as a function of stopping threshold are shown in Figure 3.5, 3.6 and 3.8 on the validating set of the PubMed, the arXiv, and the GovReport data set, respectively. The functions exhibit a local maximum between 0.1 and 1.0, which implies that when p_{thres} is too low, summaries tend to be too short, while when p_{thres} is too high, summaries will be unduly lengthy. We chose $p^*_{\text{thres}} = 0.6,\ 0.5,\ 0.8$ and 0.6 for the PubMed, the arXiv, the PubMed$_{\text{trunc}}$, and the GovReport dataset, respectively.

E Creating High-ROUGE Episodes for Training

As introduced in Section 3.4 and Algorithm 1 in the main paper, at each training iteration, we sampled a high-ROUGE episode from the set $\mathbb{E}_p$. An episode can be viewed as a sequence of state-action pairs as well as the final reward, such as $(S_0, s_{a_0}, \ldots, S_{T-1}, s_{a_{T-1}}, S_T, A_{\text{stop}}, r)$. Here, $\{s_{a_0} \ldots s_{a_{T-1}}\}$ is the extracted summary consisting of a set of T sentences, and r is the average of the associ-

ated ROUGE-1, ROUGE-2, and ROUGE-L F1 scores.

In (Nallapati et al., 2017), a greedy approach was proposed to select candidate summaries by sequentially selecting from the source document the optimal sentence that maximally improves the average ROUGE-1/2/L score once added to the current subset of selected sentences.

In this paper, we define a high-ROUGE episodes set $\mathbb{E}_p$ as the set of multiple episodes where each episode has a high average ROUGE-1/2/L F1 score. To obtain not a single episode in $\mathbb{E}_p$ but multiple episodes with high average ROUGE-1/2 scores, we modified the greedy approach by considering not only the optimal sentence at each sentence selection step but also $B-1$ sub-optimal sentences. This sentence-sampling step is repeated for each of these B new subsets to result in a potentially exponentially growing number of high ROUGE-score episodes. This process stops until no sentence can further improve the average ROUGE-1/2/L score or a maximum number N_{max} of selected sentences per episode is reached. B can be considered the branching size, analogous to beam search strategies in neural machine translation (Sutskever et al., 2014; Freitag and Al-Onaizan, 2017). We set $B = 2$ by default.

In practice, we notice that ROUGE-L F1 score is computationally intensive. Because when creating $\mathbb{E}_p$ we need to iteratively re-compute ROUGE scores once a new sentence is added to the current summary, including the ROUGE-L F1 score into computation would heavily slow down the process of creating the high-ROUGE episodes set for training. As a compromise, we do not incorporate the ROUGE-L F1 score into the intermediate steps of our modified greedy approach. Instead, we calculate the ROUGE-L F1 score only once after a complete high-ROUGE episode is selected, and use this ROUGE-L F1 score together with ROUGE-1/2 F1 scores to compute the reward r for each episode. A similar strategy was adopted in Zhou et al. (2018) to create the training dataset by maximizing ROUGE-2 F1 scores only.

We refer to an episode (S_0,s_A,S_1,s_B,S_2,s_C,S_3,A_{stop},r) as "(s_A, s_B, s_C)" for simplicity. Because permuted episodes (s_A, s_B, s_C), (s_A, s_C, s_B), and (s_C, s_B, s_A) have nearly the same average ROUGE-1/2 scores (although ROUGE-L score may

differ), we decided to equally sample them with the hope to avoid overfitting. This decision does not interfere with our usage of extraction history, because under (s_A, s_B, s_C), the agent learns to extract s_C from $\{s_A, s_B\}$, while under (s_C, s_B, s_A) it learns to extract s_A from $\{s_B, s_C\}$. Thus, history plays a role in both cases.

F Padding and Truncation of Sentences and Documents

In the training process, we used mini-batch gradient descent. To enable efficient batch-wise parallel GPU computation, each document in a mini batch needs to have the same number of sentences, and each sentence needs to have the same number of tokens. Therefore, in order to unify the sentence length to a common value L_{sen}, we appended "PAD" tokens at the end of sentences shorter than L_{sen}, and we truncated sentences longer than L_{sen}. To unify the document length in terms of number of sentences to a common value L_{doc}, we appended empty-string sentences at the end of documents shorter than L_{doc}, and truncated documents longer than L_{doc}. To ensure consistency between training and testing we also performed the same padding and truncation setting during testing. We set L_{doc} to 500 for the PubMed, the arXiv, and the GovReport datasets and 50 for the $\text{PubMed}_{\text{trunc}}$ dataset based on the document length statistics shown in Table 3.1 in the main paper. We set L_{sen} to 100 for the PubMed, the $\text{PubMed}_{\text{trunc}}$, and the GovReport datasets and 150 for the arXiv dataset, because we noticed a larger variance in the length of sentences in the arXiv dataset.

G Interactive Web Interface for Human Evaluation

To provide for a convenient evaluation procedure for volunteers, we designed an interactive web interface based on Jupyter Widgets[3]. As shown in Figure 3.9, for each document, we display the reference summary, summary A, and summary B from left to right. The reference summary contains the ground-truth abstract. Summaries A and B are the summaries extracted by the two

[3] https://ipywidgets.readthedocs.io/

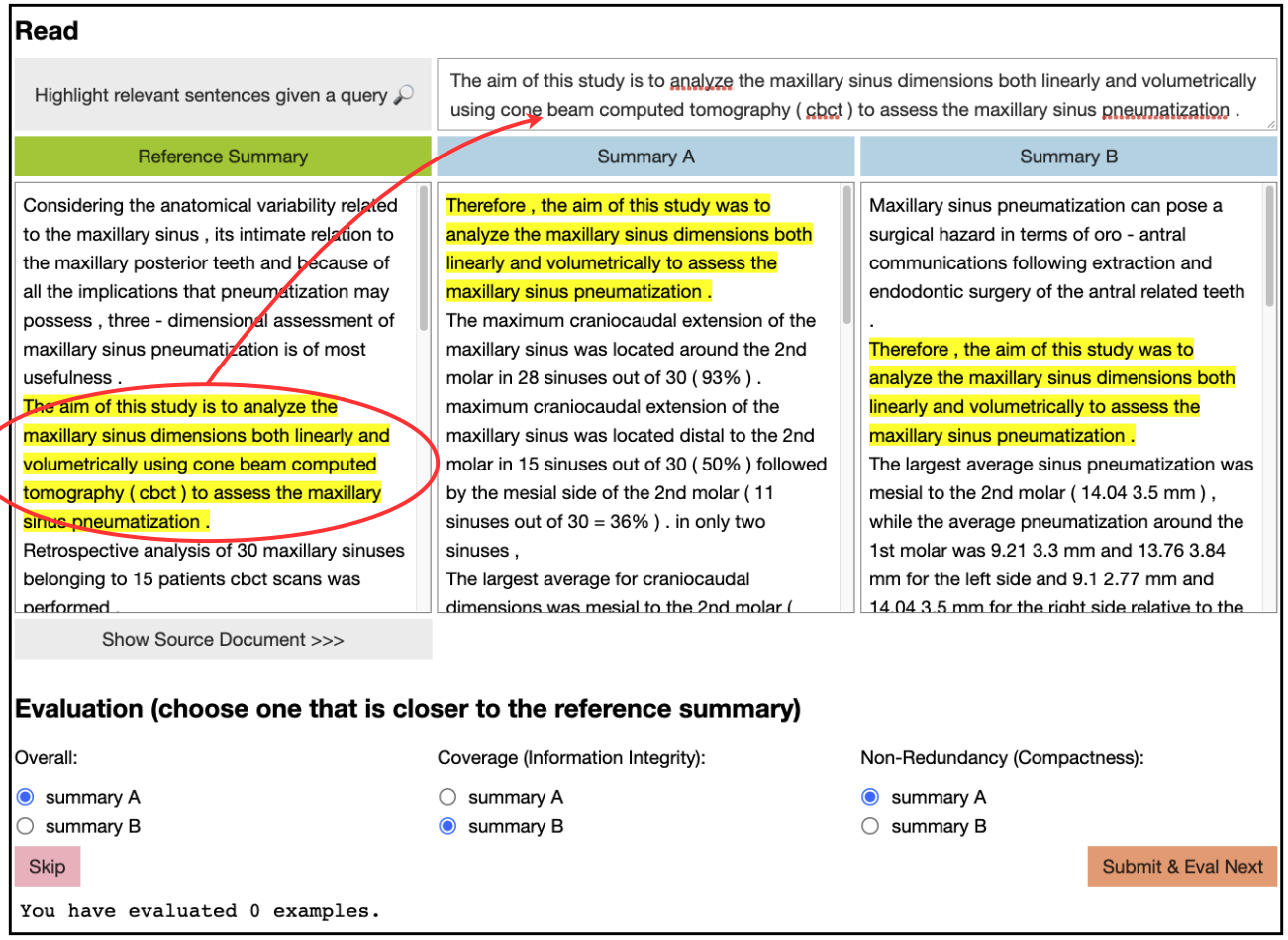


Figure 3.9: We designed an interactive web interface for the human evaluation experiments introduced in Section 3.5.4.

models assigned in a random order, so that the volunteers do not know which model either summary came from. Meanwhile, the volunteers were allowed to read the source document by clicking the button "Show Source Document >>>". We also provided a sentence highlighting function to help the volunteers rapidly retrieve relevant content. We allowed evaluators to copy a sentence from the reference summary and paste it to the text box above. After clicking the button "Highlight relevant sentences given a query", relevant sentences in both summaries were highlighted, to help the volunteers rapidly find information of interest. The relevance score of a pair of sentences was given by the cosine similarity of the two sentences' embeddings computed with Sent2vec (Pagliardini et al., 2018). In the evaluation panel the volunteers selected the better summary (A or B) by comparing the model-produced summary with the reference summary on three criteria: overall quality, coverage (in terms of information content), and non-redundancy. After making a choice they clicked the button "Submit & Eval Next" to submit the current evaluation result and evaluate the next summaries, or click "Skip" if they were not sure which summary was indeed better.

H Examples of Extracted Summaries

We provide summarization examples in Table 3.10 and 3.11. In Table 3.10, we compared MemSum trained on the arXiv dataset with Dancer Pegasus (Gidiotis and Tsoumakas, 2020) on a typical paper on which MemSum achieved higher ROUGE-1 F score than Dancer Pegasus. In Table 3.11 we provide the extractive summary of this paper itself using our MemSum model. Sentences with similar meanings in different summaries are highlighted in the same color.

I Reproducibility

The MemSum code and variants of MemSum that we used in our ablation study, as well as the MemSum parameters trained on the PubMed dataset, can be found in the submitted code.zip file. Also, we provide a sample of the

Title	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Original abstract	We introduce a new language representation model called BERT, which stands for Bidirectional Encoder Representations from Transformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial taskspecific architecture modifications. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement)
DANCER PEGASUS	Language model pre-training has been shown to be effective for improving many natural language processing tasks such as sentence-level paraphrasing and entity recognition tasks. However, current approaches to pre-trained language models are restricted to unidirectional language models. In this paper, we propose a new approach to pre-trained language models based on bidirectional encoder transformers (BERT). BERT is inspired by the pre-training objective of cloze task (Taylor et al., 1953), where the goal is to predict some masked language representations from the input. We introduce BERT and its detailed implementation in this paper. The BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Rich unsupervised pre-training is an integral part of many language understanding systems. In particular, these results enable even low-resource tasks to benefit from deep unidirectional architectures. Our major contribution is further generalizing these findings to deep bidirectional architectures, allowing the same pre-trained model to successfully tackle a broad set of NLP tasks.
ROUGE1-F1	36.52
MemSum	Language model pre-training has been shown to be effective for improving many natural language processing tasks. In this paper, we improve the fine-tuning based approaches by proposing BERT: Bidirectional Encoder Representations from Transformers. The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike Radford et al. (2018), which uses unidirectional language models for pre-training, BERT uses masked language models to enable pretrained deep bidirectional representations. BERT is the first finetuning based representation model that achieves state-of-the-art performance on a large suite of sentence-level and token-level tasks, outperforming many task-specific architectures.
ROUGE1-F1	44.29

Figure 3.10: Example summaries for Dancer Pegasus (Gidiotis and Tsoumakas, 2020) and MemSum.

Title	(This paper) MemSum: Extractive Summarization of Long Documents using Multi-step Episodic Markov Decision Processes
Original abstract	We introduce MemSum (Multi-step Episodic Markov decision process extractive SUMmarizer), a reinforcement-learning-based extractive summarizer enriched at each step with information on the current extraction history. When MemSum iteratively selects sentences into the summary, it considers a broad information set that would intuitively also be used by humans in this task: 1) the text content of the sentence, 2) the global text context of the rest of the document, and 3) the extraction history consisting of the set of sentences that have already been extracted. With a lightweight architecture, MemSum obtains state-of-the-art test-set performance (ROUGE) in summarizing long documents taken from PubMed, arXiv, and GovReport. Ablation studies demonstrate the importance of local, global, and history information. A human evaluation confirms the high quality and low redundancy of the generated summaries, stemming from MemSum's awareness of extraction history.
MemSum summary	In this paper, we propose to model extractive summarization as a multi-step episodic Markov Decision Process (MDP). As shown in Figure 1, at each time step in an episode, we define a sentence state composed of three sub-states: 1) the local content of the sentence, 2) the global context of the sentence within the document, and 3) information on the extraction history, including the previously selected set of unordered sentences and the remaining sentences. To efficiently encode local and global sentence states, we design an extraction agent based on LSTM networks. We show that extraction-history awareness allows our model to extract more compact summaries than models without history awareness and behave more robustly to redundancies in documents. 3) Our model outperforms both extractive and abstractive summarization models on PubMed, arXiv, and GovReport datasets
ROUGE1-F1	48.57

Figure 3.11: MemSum summary of this paper.

datasets used in this paper in the data.zip file, as well as the raw data for the human evaluation. This will ensure that the results in this work are well reproducible.

Chapter 4

Controllable Citation Text Generation

Originally published as: Gu, Nianlong and Richard Hahnloser. "Controllable Citation Text Generation." arXiv preprint arXiv:2211.07066 (2022).

Abstract

The aim of citation generation is usually to automatically generate a citation sentence that refers to a chosen paper in the context of a manuscript. However, a rigid citation generation process is at odds with an author's desire to control the generated text based on certain attributes, such as 1) the citation intent of e.g. either introducing background information or comparing results; 2) keywords that should appear in the citation text; or 3) specific sentences in the cited paper that characterize the citation content. To provide these degrees of freedom, we present a controllable citation generation system. In data from a large corpus, we first parse the attributes of each citation sentence and use these as additional input sources during training of the BART-based abstractive summarizer. We further develop an attribute suggestion module that infers the citation intent and suggests relevant keywords and sentences that users can select to tune the generation. Our framework gives users more control over generated citations, outperforming citation generation models with-

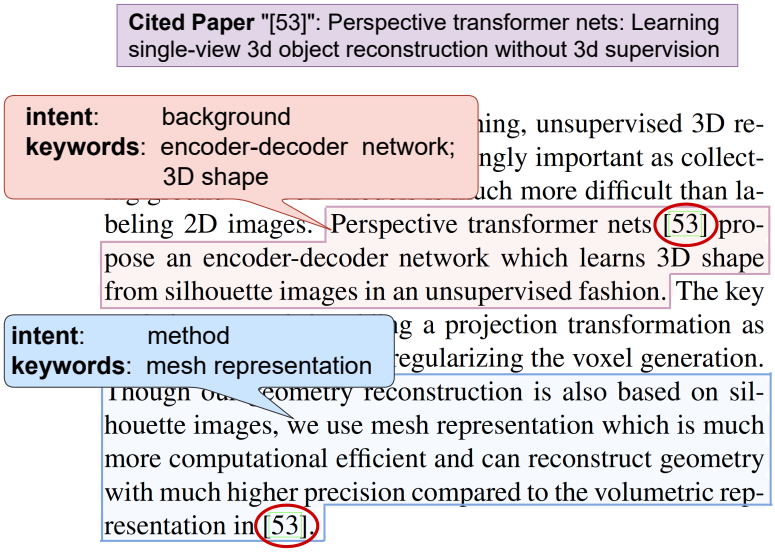


Figure 4.1: Example citation sentences of the same paper, taken from a paragraph in (Liu et al., 2019). The citations differ in intents and keywords.

out attribute awareness in both ROUGE and human evaluations.

4.1 Introduction

A common practice in scientific writing is to cite and discuss relevant papers in support of an argument, in provision of background information, or in comparison of results (Penders, 2018). Recent studies aim to automate this process by using neural networks to generate a citation sentence based on information from the manuscript being written and/or the paper to be cited. For example, Nikiforovskaya et al. (2020) proposed a BERT-based extractive summarizer (Liu, 2019) that produces a paper review by extracting one sentence from each of the related papers. Chen and Zhuge (2019) proposed to automatically generate a related work section by extracting information on how papers in the reference list have been cited in previous articles. Xing et al. (2020) developed a RNN-based pointer generator network that can copy words from the manuscript and the abstract of the cited paper based on cross-attention, which was further extended in (Ge et al., 2021) using information from the citation graph to enhance citation generation.

These efforts focused primarily on developing fully automated pipelines and they left little room for users to control the generation process. However, we believe control is desirable for the following considerations. Authors often have a clear motivation before writing a citation sentence. For example, they may have a specific *intent* to cite, such as comparing results or presenting background information; they may have *keywords* in mind to appear in the citation sentence; or they may want to refer to a particularly *relevant sentence* in the body text of the cited paper, e.g., a specific experimental finding. Even in a given context, the motivation to cite a paper can be diverse (Figure 4.1). When the generated citations do not match an author's motivation, the author may wish to change the generation by specifying certain attributes, such as citation intent, keywords, or relevant sentences. However, recent works do not allow for this possibility (Wang et al., 2022; Xing et al., 2020) because their generation process is not conditional on these properties.

In order to allow users to freely adjust the generation, we propose a controllable citation generation system consisting of two modules: a conditional citation generator and an attribute suggester.

The conditional citation generator is based on BART-large (Lewis et al., 2020), a transformer-based (Vaswani et al., 2017) text denoising autoencoder. During the training period, we let BART-large receive two sets of input sources. The first group is **contextual text**, including 1) the sentences preceding the target citation sentence as *local context* in the manuscript and 2) the title and abstract of the cited paper as *global context*. The second group is formed by the **attributes** associated with the target citation sentence, including the *citation intent*, the *keywords*, and the most *relevant sentences* in the body text of the cited paper. We add citation attributes as input to make the decoding process of BART-large conditional on these attributes.

The citation attribute suggestion module modifies and fine-tunes SciBERT (Beltagy et al., 2019) to infer possible intents and suggest keywords and related sentences based on the contextual text and the body text of the cited paper. The purpose of this module is to propose to a user attributes to guide the generation of citation sentences. We believe that such a suggestion-selection-

generation pipeline can maintain a fair degree of automation while retaining good controllability.

Our contributions are summarized as follows:

- We propose a controllable citation generation pipeline with an attribute suggestion module and a conditional citation generation module;
- We evaluate the controllability of our system in response to different attributes using automated metrics and human evaluation;
- We parse the contextual text and citation attributes of each citation sentence to build a large dataset that can be used for future studies on controllable citation generation.

4.2 Related Work

Our idea of a controllable citation generation system originated from the study of conditional generative models that were proposed in Sohn et al. (2015) and applied to controlled text generation tasks (Ficler and Goldberg, 2017; Keskar et al., 2019; Dathathri et al., 2020). For example, Keskar et al. (2019) used attributes (or control codes) such as genre or topic as conditions when pretraining an autoregressive language decoder based on self-attention (Vaswani et al., 2017) to encourage the next-token prediction (during decoding) conditional on a specific attribute. In this paper, we apply this conditional approach to the citation generation task, where we introduce a set of citation-related attributes as conditional input text when training an encoder-decoder model to generate citation sentences.

Recently, Jung et al. (2022); Wu et al. (2021) proposed intent-controlled citation generation models that focus on controlling the intention of generated citations, while in this paper, we deal with more conditions beyond the intent of the citation. In addition, we propose solutions to automatically suggest potential attributes to better balance the automation and controllability of citation generation.

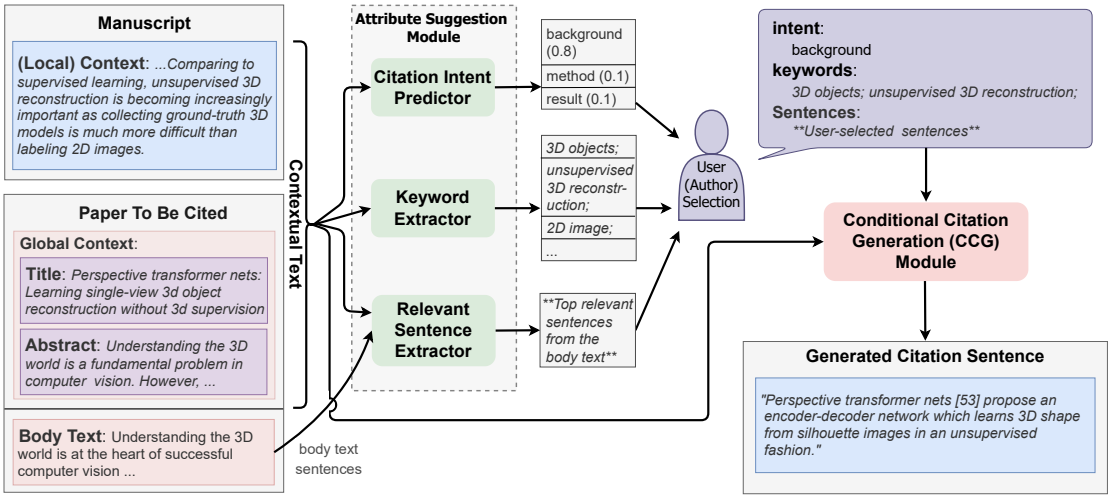


Figure 4.2: The pipeline for our controllable citation text generation system. The attribute suggestion module suggests candidate attributes, including citation intent, relevant keywords, and sentences. Users can select the desired attributes from the suggestions and use them as conditions to guide the citation generation module.

4.3 Model

We introduce our controllable citation generation system (outlined in Figure 4.2) in this section.

4.3.1 Conditional Citation Generation Module

The **C**onditional **C**itation **G**eneration (CCG) module outputs a citation sentence Y given the contextual text X and the citation-related attributes C. C include 1) citation intent, 2) keywords, and 3) sentences in the body text that are related to the citation sentence. The training objective is to maximize the conditional generation probability:

$$p(Y|X,C) = \prod_{i=1}^{n} p(y_i|y_1 y_2 \dots y_{i-1}, X, C), \tag{4.1}$$

where y_i is the i_{th} token in the citation sentence Y. This task can be modeled as a sequence-to-sequence problem where we consider X and C as input text, Y as output sequence, and train BART-large to minimize the negative log-likelihood $-\log p(Y|X,C)$. During testing, we let BART-large take the contextual information X and the attribute C as inputs to the encoder and generate

the citation sentence on the decoder side by next-token prediction to maximize the probability in Equation (4.1).

For X, we define the local context in X as up to N_s ($N_s = 5$) sentences (from the same section) that precede the target citation sentence to be generated. The global context in X is formed by the title and abstract of the cited paper. For C, when training the citation generator, we parse the ground truth citation sentence to obtain the oracle citation attributes (Section 4.4.3). During testing, the attributes C are either recommended by our attribute suggestion module or specified by the users.

To allow the BART-large encoder to distinguish between different input sources, we add a special field name (or control code) before each input source, following Raffel et al. (2020); Keskar et al. (2019). The final input text is structuted as: "**intent:** [Citation intent] **keywords:** [Keywords relevant to the citation, separated by ';'] **sentences:** [Body text sentences relevant to the citation] **context:** [local context in the manuscript] **title:** [Title of the cited paper] **abstract:** [Abstract of the cited paper]".

4.3.2 Attribute Suggestion Module

The attribute suggestion module consists of three submodules: a citation intent predictor, a keyword extractor, and a relevant sentence extractor.

The Citation Intent Predictor predicts the intent of a citation sentence based on the local context in the manuscript and the global context (title and abstract) of the cited paper. Following Cohan et al. (2019), we consider three possible intents i: 1) *background*, summarizing related work or concepts as background; 2) *method*, using a certain method or dataset (e.g., the second citation sentence in Figure 4.1); and 3) *result*, comparing results.

We connect the local and global context texts with a special token "[SEP]" and prepend a token "[CLS]" at the beginning. We use SciBERT to compute from the concatenated text the last hidden states of all tokens. We then input the last hidden state of "[CLS]" into the intent prediction header, a fully connected two-layer network to obtain the output $x_{\text{intent}}(i)$. Finally, we apply the softmax

to $x_{\text{intent}}(i)$ to produce the likelihood $p_{\text{intent}}(i)$ for three possible intents. After obtaining the ground truth intent i_{true} by parsing the target citation sentence (Section 4.4.3), we train SciBERT and the intent prediction header to minimize the cross-entropy loss:

$$\mathcal{L} = -\frac{\exp\left(x_{\text{intent}}(i_{\text{true}})\right)}{\sum_{i \in \text{all intents}} \exp\left(x_{\text{intent}}(i)\right)} \tag{4.2}$$

The Keyword Extractor extracts keywords relevant to the target citation sentence from the contextual text. We obtain a set of candidate keywords by chunking all noun phrases[1] in the contextual text. We embed the contextual text q into a query embedding v_q and each candidate keyword k_i into a keyword embedding v_{k_i} using the same SciBERT. The text encoding is done by averaging the last hidden states of all tokens in the text. Then we rank the keywords based on the cosine similarity between v_{k_i} and v_q. Inspired by Zhong et al. (2020), we fine-tune the SciBERT encoder so that relevant keywords are ranked at the top positions. Here, we measure relevance using the average of the ROUGE-1&2 F1 scores between the keyword and the target citation sentence, and we assign a rank to each keyword based on its relevance score. Higher relevance scores correspond to smaller rank values (minimum of 1), and keywords with the same relevance score are assigned the same rank. Within the candidate keyword list, given a pair of keywords k_i and k_j whose ranks r_i and r_j satisfy $r_i < r_j$, we fine-tune SciBERT with the triplet loss:

$$\begin{aligned} \mathcal{L} = \max(0, f(v_q, v_{k_j}) - f(v_q, v_{k_i}) \\ +(r_j - r_i) \times \gamma)), \quad r_i < r_j, \end{aligned} \tag{4.3}$$

where $f(v_a, v_b) = \frac{v_a^T v_b}{\|v_a\| \|v_b\|}$ denotes the cosine similarity between the vectors (v_a, v_b) and γ is the margin used in the triplet loss (Schroff et al., 2015).

In order to extract relevant and diverse keywords, the keyword extractor selects keywords based on maximal marginal relevance (MMR) (Carbonell and

[1]We use the noun phrase chunker from Spacy: https://spacy.io/ and remove the articles of the noun phrases.

Goldstein, 1998): at each step, it picks an unselected keyword k_i that maximizes:

$$(1-\alpha) f(v_q, v_{k_i}) - \alpha \max_{k_j \in S_k} f(v_{k_i}, v_{k_j}), \tag{4.4}$$

where α is the diversity factor and S_k represents the set of selected keywords. We set $\alpha = 0.2$ to reduce redundancy among extracted keywords.

The Sentence extractor extracts the sentences most relevant to the target citation sentence from the body text of the cited paper. Similarly to the keyword extractor, we use a SciBERT text encoder to encode contextual text and body sentences as embeddings. We then use the contextual text as a query and rank the body sentences based on the cosine similarity between the query embedding and the sentence embedding. We fine-tune the SciBERT encoder to encourage sentences that are more relevant to target citation sentences (as measured by the average ROUGE-1 and ROUGE-2 F1 scores) to rank in the top positions, the same strategy used to fine-tune the keyword extractor. Furthermore, when calculating the triplet loss similar to Equation (4.3), we clamp the rank value of the sentences to 10: $r_i = \min(r_i, 10)$, because we focus only on the ranking order of the most relevant sentences.

4.4 Dataset Preparation

4.4.1 Cited Paper Filtering

For the training set, we filter a subset of S2ORC (Lo et al., 2020) papers that were published between 2000 and 2020 in the domains of biology, medicine, or computer science. Furthermore, we select only papers that are cited in at least 50 sentences contained in S2ORC. With this criterion, we obtain for each paper many citation sentences, hopefully with diverse citation attributes (intents and keywords). Training our conditional citation generator on such a dataset likely allows the generator to learn to cite papers under diverse conditions.

For the validation and the test sets, we obtain citation sentences by parsing papers from arXiv (Kaggle, 2022) and PMCOA (of Medicine, 2003) that were

published in 2022. We do not filter cited papers based on the number of citations, because we want to test our system in a real-world scenario where cited papers vary in terms of domain and number of citations.

4.4.2 Citation Sentence Filtering and Train/Test Decoupling

In the citation generation dataset proposed in Xing et al. (2020), some citation sentences citing more than one paper appeared in both the training and the test sets. For example, the following sentence "*co-occurrence of words [Paper A] and discourse relations [Paper B] also predict coherence.*" appeared as the target citation sentence in two samples, one in the training set and the other in the test set. It follows that the local context in the manuscript is identical for these two data samples. The only difference is that in the training sample paper A is considered as the cited paper, while in the testing test paper B is treated as the cited paper.

First, we argue that it is an ill-defined problem to train a model to generate a citation sentence that should cite two papers when only one paper is provided as the global context. Second, the fact that some samples from the training and test sets are coupled increases the chance of overfitting: A coupled pair of training and testing samples share the same local context as the input and the same citation sentence as the target output. Thus, the citation generation model can "remember" and thus "recite" the corresponding citation sentence during testing given a local context that has been used during training.

To this end, when creating our **sci**entific **c**ontrollable **c**itation **g**eneration (SciCCG) dataset, we eliminated citation sentences that were too short (<5 words) or too long (>200 words), or those that cited more than one paper. In addition, we split the training, validation, and test sets with the following decoupling rule: Given a citation sentence s from one set (e.g. training set), we refer to the paper from which s comes as the citing paper $p_{s,\text{citing}}$ and the paper cited by s as the cited paper $p_{s,\text{cited}}$. When splitting the training, validation, and test sets, we ensure that the citation sentences from the other two sets neither come from $p_{s,\text{citing}}$ nor do they cite $p_{s,\text{cited}}$.

Intent Category (# samples)	Background (1,014)	Method (613)	Result (260)	Average (Macro)
Jurgens et al. (2018)	84.7	74.7	78.2	79.2
Cohan et al. (2019)	87.8	84.9	79.5	84.0
SciBERT+scaffolds (Ours)	**89.1**	**87.1**	**84.0**	**86.7**

Table 4.1: The F1 scores on three citation intent categories and the avarage (macro) F1, tested on the SciCite dataset created by Cohan et al. (2019).

Information	Training	Validation	Test
# cited papers	59,645	1,373	1,404
# citing papers	1,404,690	1,360	1,382
# citation sentences	2,678,983	1,385	1,411
Statistics of parsed citation attributes of the citation sentences			
# *background* intent	2,188,066	1,075	1,093
# *method* intent	388,437	213	219
# *result* intent	102,480	97	99

Table 4.2: The statistics of our SciCCG dataset.

4.4.3 Parsing Citation Attributes

Training the conditional citation generator (Section 4.3.1) and the citation intent predictor (Section 4.3.2) requires citation attributes as labels. However, the true attributes of the citation sentences are usually not explicitly provided by the authors. For training we therefore use pseudolabels by inferring citation attributes from the target citation sentences.

Citation Intent. We infer the intent of citation sentences using a SciBERT-based intent classifier that has the same structure as our citation intent predictor, except that it differs in terms of input text and purpose. The intent predictor takes contextual text as input and aims to predict the citation intent before the citation sentence is written. In contrast, the intent classifier works as a tutor. It takes the actual citation sentence as input and infers the true intent, which can be used to "teach" the citation intent predictor to predict the most likely intent.

We use a multitask training strategy (Cohan et al., 2019) to train the intent classifier. In addition to the main task of classifying citation intent, we add

Mode	Citation Generation Model	Citation Attributes			Performance		
		citation intent	relevant keywords	relevant sentences	R-1	R-2	R-L
fully automatic	PTGEN-Cross†	–	–	–	24.95	4.12	18.52
	BART-large	–	–	–	29.62	7.56	22.51
	BART-large-CCG	–	–	–	29.43	7.53	22.70
		Random intent predictor (ours)	–	–	28.16	6.57	21.37
			–	–	29.32	7.26	22.52
		–	KeyBERT keyword extractor (ours)	–	29.46	6.71	21.78
		–		–	30.58	7.78	22.87
		–	–	Sentence-BERT sentence extractor (ours)	29.87	7.08	21.93
		–	–		30.91	7.76	**23.02**
		intent predictor	keyword extractor	sentence extractor	**31.31**	**7.98**	22.92
user-controlled	BART-large-CCG	Ground Truth	–	–	29.85	7.75	23.10
		–	Ground Truth	–	38.74	12.53	28.68
		–	–	Ground Truth	36.29	12.28	27.52
		Ground Truth	Ground Truth	Ground Truth	**42.89**	**15.91**	**31.61**

Table 4.3: We evaluated the performance of citation generation in a fully automated mode and a user-controlled mode. We used the same trained BART-large-CCG model for all evaluation tasks. † represents our own implementation. The symbol "–" means that the corresponding citation attribute was not used as the conditional input for generation.

two auxiliary classification tasks (scaffolds) (Cohan et al., 2019) to improve the performance on the main task: 1) predicting the title of the section to which the cited sentence belongs; and 2) detecting whether the sentence needs a reference (citation worthiness). For each auxiliary task, we use a separate functional head, a two-layer fully connected network, with the SciBERT-encoded "CLS" hidden states as input, to classify section titles and citation worthiness, respectively. The training loss is a weighted sum of the cross-entropy losses (Equation (4.2)) of the three tasks, with a weight of 1.0 for the main task, and weights of 0.05 and 0.01 for the auxiliary tasks 1) and 2), respectively.

We use our SciBERT-based intent classifier (performance shown in Table 4.1) to parse the intent of each citation sentence in the training/validation/test set and use it as the ground truth citation intent of the target citation sentence when training the intent predictor and and the conditional citation generator.

Relevant Keywords. After chunking all noun phrases in the contextual text as candidate keywords, our goal is to select a set of keywords that as a whole have the highest ROUGE-1 & 2 F1 scores compared to the target citation sentence. Following the greedy selection strategy (Gu et al., 2022a), we select a keyword whose addition to the already selected set of keywords maximally increases the ROUGE F1 scores, and this selection process stops when the ROUGE F1 scores no longer increase or at least 3 keywords have been selected. Greedily selected keywords are used as the ground truth relevant keywords of the target citation sentence when training the CCG module.

Relevant Sentences. We extract up to 2 sentences from the body text of the cited paper using the same greedy strategy as for extracting relevant keywords.

The statistics of our dataset are shown in Table 4.2.

4.5 Experiment

We choose a learning rate of 1e-5 to fine-tune the modules. When training the CCG module, we truncated the input sequence to a maximum length of 512

tokens and truncated the output to 75 tokens. To make the CCG module robust to the unavailability of conditional citation attributes, we randomly drop certain citation attributes during training as follows: 1) We set the citation intent to the empty string with probability 0.5; 2) Given n relevant keywords/sentences of the target citation sentence, we randomly select m ($0 \leq m \leq n$) keywords/sentences and use them as the conditional input.

For the citation intent predictor, the size of the hidden layer of the intent prediction head is 32. We observed that the categories of citation intent were unbalanced and most of the citation sentences were "background" intents (Table 4.2). Therefore, when training the citation intent predictor, we downsampled the "background" citation sentences to 1/5 of the original number to balance the number of samples for all intent categories. For training the keyword extractor and the sentence extractor, we used the margin $\gamma = 0.01$ in the triplet loss (Equation (4.3)). We evaluated the generated citation sentences using ROUGE-1,2 and L (Lin, 2004) F1 scores.

4.6 Results and Discussion

We evaluated the performance of the citation generation in both the fully automatic mode and the user-controlled mode.

In the fully automatic mode, we assume that the ground truth citation attributes are unknown. We adopted a *suggestion-generation* strategy to automatically generate citation sentences. We first use the attribute suggestion module to suggest citation attributes (citation intent, 3 relevant keywords, and 2 relevant sentences) and then let BART-large-CCG take contextual text and suggested attributes as input and generate citation sentences. We compared our method with 1) PTGEN-Cross (Xing et al., 2020) and 2) BART-large that were both trained without using citation attributes as conditions. In addition, we compared our attribute suggestion module with the following baselines: 1) randomly selecting one intent from all three intents; 2) extracting keywords using KeyBERT[2] (Grootendorst, 2020) and 3) ranking and extracting body

[2]We used the BERT emebedding model "all-mpnet-base-v2" and set the diversity to 0.2

text sentences based on embeddings encoded by Sentence-BERT (Reimers and Gurevych, 2019).

In the user-controlled mode, we assume that ground truth citation attributes are available. This happens when users want to control the generation by specifying the desired citation attributes.

4.6.1 Results Comparison

In automatic mode, BART-large-CCG taking citation attributes suggested by our attribute suggestion module as conditional input outperformed unconditional generation models, including BART-large and PTGEN-Cross (Table 4.3). For example, when we let BART-large-CCG take as input 1) contextual text and 2) keywords extracted also from the contextual text using our keyword extractor, BART-large-CCG achieved a higher ROUGE score than BART-large, even though both methods take the contextual text as the only source of information. We observed that BART-large-CCG performed best when it used all the citation attributes (intent, keywords, and sentences) suggested by our attribute suggestion module. These results show that our pipeline for generating citation sentences using automatically suggested citation attributes is effective without human guidance.

In addition, our SciBERT-based keyword extractor outperformed KeyBERT when used as a relevant keyword suggestion module, and the SciBERT-based sentence extractor outperformed Sentence-BERT when suggesting relevant body sentences. We attribute this improvement to the fine-tuning process using triplet loss, which allows the SciBERT text encoder to better estimate the similarity between contextual text and keywords/sentences.

In the user-controlled mode, the performance of BART-large-CCG improved when we added ground truth citation attributes as conditional input, especially when the generation was conditioned on relevant keywords or relevant body text sentences from the cited paper. And the best performance was observed when all three attributes were used as conditional inputs. The large

when using MRR.

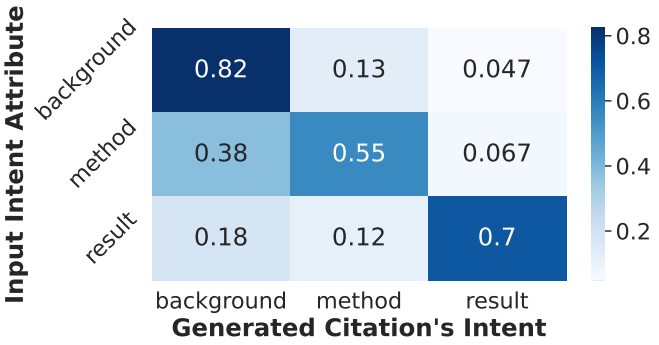
performance gain over the fully automatic mode indicates that the citation attributes are crucial for generating good citation sentences and should be considered as conditional input whenever available, for example, when authors want to cite a paper with specific keywords or relevant sentences they want to discuss.

The results in Table 4.3 show that our BART-large CCG model has stable performance regardless of whether attributes are suggested by our attribute suggestion module or specified by users, and is flexible and robust under different availability of citation attributes, indicating good controllability, which we will investigate further next.

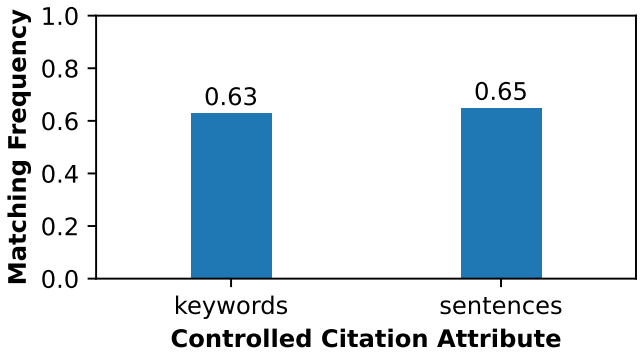
4.6.2 Controllability of CCG Module

We tested whether the generated citation sentences reflect the provided conditional attributes as follows. For each test sample, we manually assigned a conditional intent (”background”, ”method”, or ”result”) and let BART-large-CCG use the assigned intent attribute, leaving the other conditional attributes (e.g., relevant keywords and sentences) and the contextual text unchanged during generation of a citation sentence. We then determined the intent of the generated citation using our SciBERT-based intent classifier (Section 4.4.3). We calculated the intents category frequencies of the generated citations and plotted the confusion matrix (Figure 4.3a). The large values of the diagonal elements in the confusion matrix imply that the intents of the generated citations tend to be consistent with the desired intent provided as input, indicating that our model effectively adapts the generated text to the desired intent.

Unlike for citation intent, for relevant keywords and sentences there is no fixed number of pre-defined categories. For each test sample, we randomly selected two keywords A and B from the first 5 keywords extracted by our SciBERT-based keyword extractor. We first used each keyword separately as a condition to guide the generation of BART-large-CCG. We then determined the keyword (A or B) that was semantically closer to the generated citations by calculating the cosine similarity between the embeddings (encoded by Sentence-BERT



(a) Citation intents as controlled attributes.



(b) Relevant keywords and sentences as controlled attributes.

Figure 4.3: We evaluated the controllability by the matching frequency between the controlled input attributes and the attributes of the generated citation sentences.

(Reimers and Gurevych, 2019)) of the two keywords and those of the generated citation sentences. We calculated the frequency of the generated citation sentences being semantically closer to the associated keywords, e.g., the citation sentence generated with keyword A as a condition was closer to keyword A than to B. Similarly, to test the controllability by sentence attribute, we randomly selected two sentences from the first 5 relevant sentences extracted by the SciBERT-based sentence extractor and conducted the same experiment. The results in Figure 4.3b show a good semantic match between the conditional keywords (or conditional sentences) and the generated citation sentences, indicating that the model effectively makes use of keywords and related sentences to guide the CCG generation process.

cited title: Low COVID-19 Vaccine Acceptance Is Correlated with Conspiracy Beliefs among University Students in Jordan **cited abstract:** Vaccination to prevent coronavirus disease 2019 (COVID-19) emerged as a promising measure to overcome the negative ... The intention to get COVID-19 vaccines was low: 34.9% (yes) compared to 39.6% (no) and 25.5% ... **context in manuscript:** A third of the entire sample notes that they do not plan to vaccinate, another third doubts the decision and focuses on the more distant results of the vaccination program conducted in the country, 11.6% are already vaccinated, and 13.3% plan to vaccinate shortly. ***¿¿generate a citation sentence HERE¡¡***
[background][acceptability] The acceptability of the COVID-19 vaccine in Jordan is low, with only 34.9% of respondents stating that they intend to vaccinate [].
[method][questionnaire] The intention to vaccinate was assessed using the COVID-19 vaccine hesitancy questionnaire [].
[result][COVID-19 vaccines] This finding is in line with the results of a previous study conducted in Jordan, which showed that the intention to get COVID-19 vaccines was low: 34.9% (yes) compared to 39.6% (no) and 25.5% (maybe) [].

Table 4.4: Generated citation sentences are guided by the citation intent and the keywords provided.

These results show that our CCG module has good controllability, making it suitable for controlling the generated citations with conditional attributes (see an example in Table 4.4).

4.6.3 Human Evaluation

We performed a human evaluation to test whether our BART-large-CCG-based controllable citation generation pipeline produces more satisfactory citation sentences compared to BART-large, which cannot be controlled by citation attributes. In the Web interface[3] we created for human evaluation of a test sample (see Appendix B), we first provide the context text in the manuscript and the content of the cited paper (title, abstract, and body). In addition, we use our attribute suggestion module to suggest the citation intent, top 5 relevant keywords, and the top 5 relevant body sentences. Participants were given the freedom to select suggested attributes and contribute their own to guide the generation of citations if they deemed it necessary.

[3]Our human evaluation website is at: https://ccg-human-eval.vercel.app/

Metric	BART-large-CCG	Neutral	BART-large
Informative	**44.44***	28.89	26.67
Coherent	**51.11***	17.78	31.11
Intent-Matched	**51.11***	26.67	22.22

Table 4.5: Results of human preference ratings for sentences produced by the three methods (in %). "Neutral" means no preference for sentences generated by either model. "*" represents statistical significance, $p < 0.05$.

We then show two sentences: 1) the sentence generated by BART-large using only contextual text as input; and 2) the sentence generated by our BART-large-CCG using contextual text and user-specified attributes as input. These sentences were presented in random order to prevent participants from identifying the method behind the sentences based on the order of appearance. Inspired by Zhang et al. (2021), we asked participants to report their preferences among the two sentences based on the creiteria 1) **informative**, whether the sentence is informative and faithful; 2) **coherent**, whether the sentence is logical and consistent with previous sentences in the manuscript; and 3) **intent-matched**, whether the sentence matches the tester's citation intent (or purpose).

Human evaluation results (Table 4.5) showed that our BART-large-CCG conditional on citation attributes was preferred compared to the unconditional BART-large model. Compared with BART-large, BART-large-CCG generated sentences with higher coherence and more informative contents. The higher value of the "Intent-Matched" metric reflects the good controllability of the BART-large-CCG given the user-specified attributes. Additionally, we observed that participants selected on average 1.8 suggested keywords and 1.6 suggested sentences as attributes for citation generation, indicating that our attribute suggestion module holds promise for use in controllable citation generation systems.

4.7 Conclusion

We proposed a controllable citation generation framework that consists of a citation attribute suggestion module and a conditional citation generation module. Our framework not only outperforms previous uncontrolled approaches in fully automated generation mode, but also manifests good control of generated citations by reflecting user-specified attributes. Our approach allows users to efficiently select the suggested citation attributes to guide the generation process, thus giving the generated sentences a better chance of reflecting a user's intentions during scientific writing. Moreover, we filter out ambiguous citation sentences and decouple the training and test sets, which makes our SciCCG dataset a good testbed for future controllable citation generation studies.

4.8 Appendices

A Computing Hardware

We train the models using 8x NVIDIA GeForce RTX 3090 GPUs. During test and evaluation, we used an NVIDIA RTX A6000 48GB GPU.

B Web Interface for Human Evaluation

We designed a user-friendly web interface to allow participants to evaluate our controlled citation generation pipeline and to compare citation sentences generated by our controlled BART-large-CCG model with those generated by the uncontrolled BART-large model. As shown in Figure 4.4, the content in the web page is divided into two columns. The left column contains the contextual text, including the context in the manuscript as well as the title and abstract of the cited paper. Users can also view the body of the text by clicking on the button at the bottom right.

In the right column of the page, we first provide the citation attributes suggested by our attribute suggestion module, which include 1) the top 5 relevant

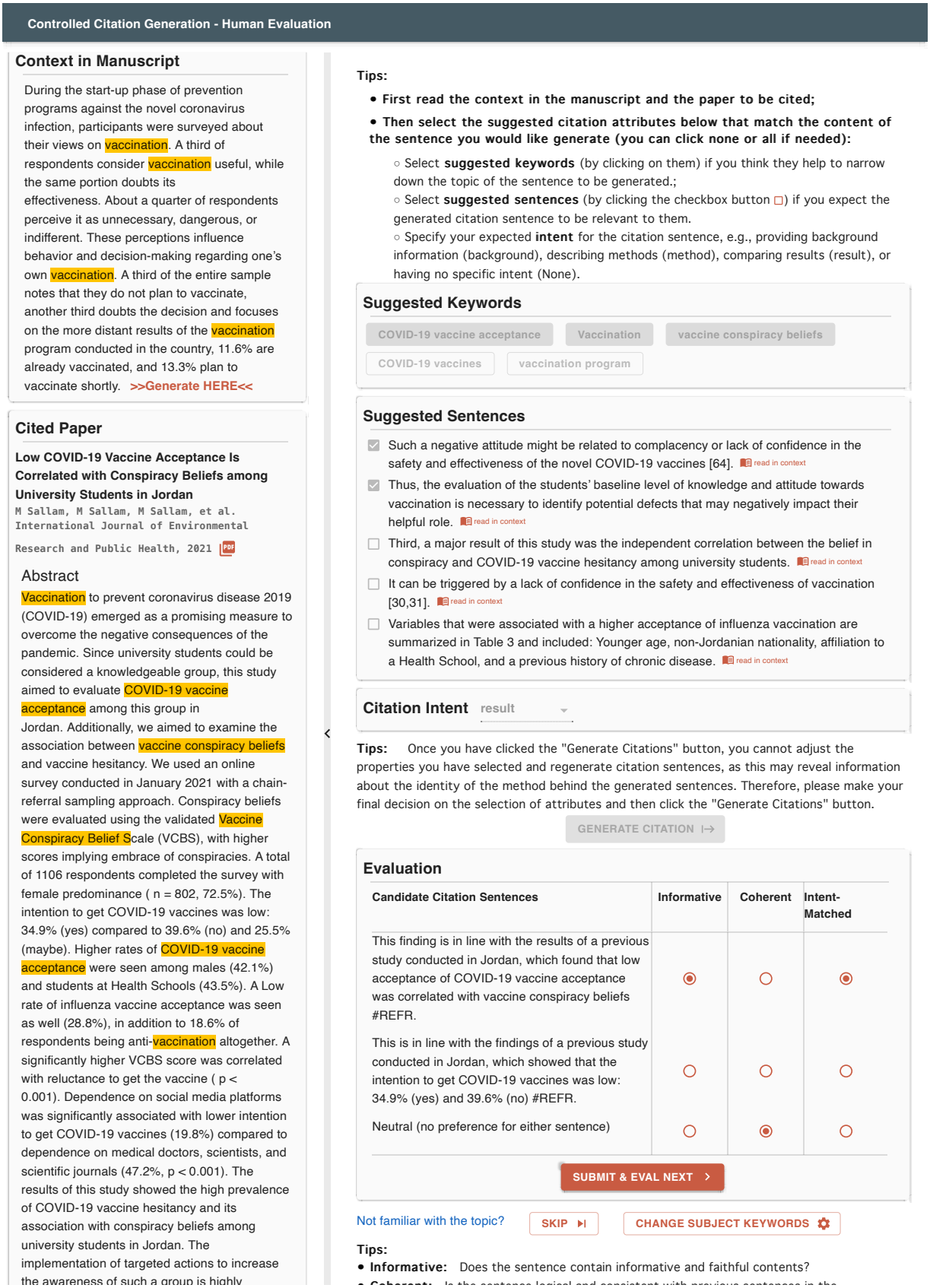


Figure 4.4: We designed an interactive web interface that allowed participants to evaluate and compare citation sentences generated by our controllable BART-large CCG model with those generated by the non-controllable BART-large model.

keywords, 2) the top 5 relevant sentences, and 3) the most likely citation intent. From these suggested attributes, users can select keywords and sentences or modify the citation intent, all in an intuitive way with a simple click of a button.

In addition, to help users select citation attributes, we highlight user-selected keywords in the contextual text, and make it easy for users to locate the suggested sentence in the body text of the cited paper so that they can better understand the sentence through a richer context.

Note that the user can choose not to actively specify any citation attributes. In this case, our BART-large-CCG model will simply use the machine-suggested attributes (top 3 suggested keywords, top 2 suggested sentences and the suggested intent) as the conditions for generation, just like the fully automatic model described in this section 4.6.

Once users click the "Generate Citation" button, we will present two sentences in a random order. One of the sentences is generated by our BART-large-CCG model, which takes as input the contextual text and the attributes chosen by the user, while the other sentence is generated by BART-large, which takes as input only the contextual text. In addition, we provide a third option, "neutral", in case the user has no preference for any of the above sentences. After users have selected their preferences in the three areas (informative, coherent, and intent-matched), they can click the "Submit & Eval Next" button to submit their evaluation results and evaluate the next test example.

In addition, we allow participants to narrow down the research area of the examples to be evaluated by specifying some subject keywords, e.g. "machine learning; translation". We will use the subject keywords as queries to retrieve relevant test examples (using BM25) and present them to the participants. In this way, participants can evaluate examples with which they are familiar, which we believe will increase the credibility of the evaluation results.

Chapter 5

SciLit: A Platform for Joint Scientific Literature Discovery, Summarization and Citation Generation

Abstract

Scientific writing involves retrieving, summarizing, and citing relevant papers, which can be time-consuming processes. Furthermore, in many workflows, these processes are serially linked, which offers opportunities for natural language processing (NLP) to provide end-to-end assistive tools. We propose SciLit, a pipeline that automatically recommends relevant papers, extracts highlights of a paper and suggests a reference sentence to cite the paper based on the user-provided context and keywords. Based on the latter, SciLit efficiently recommends papers from large databases of hundreds of millions of papers using a two-stage pre-fetching and re-ranking literature search system that allows simple plugging and unplugging of paper databases. We provide a convenient user interface that displays the recommended papers as extractive summaries and that offers abstractively-generated citing sentences that align with the context and mention the chosen keyword(s).

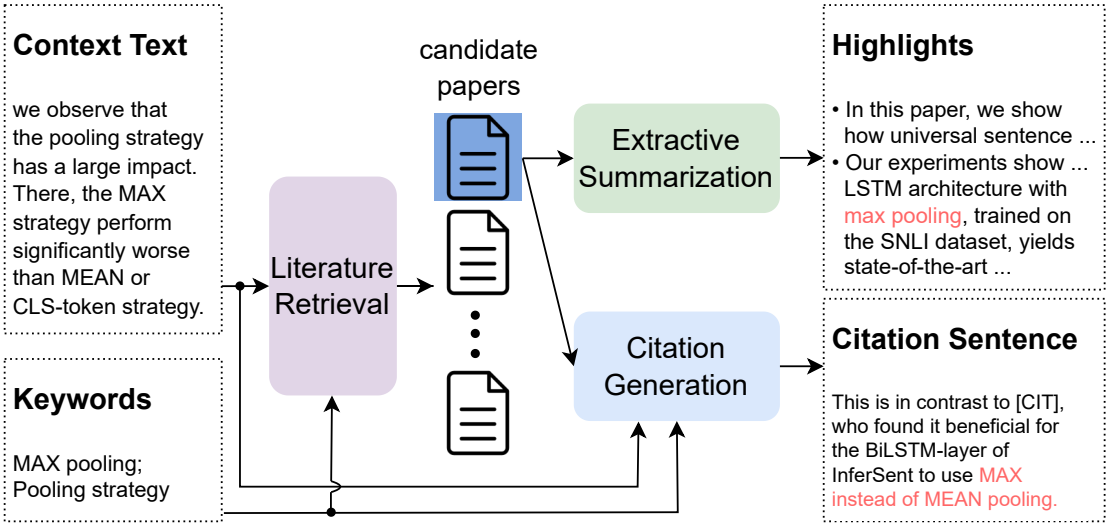


Figure 5.1: The main workflow of our platform.

Our assistive tool for literature discovery and scientific writing is available at https://scilit.vercel.app.

5.1 Introduction

When we compose sentences like "Our experiments show that XXX performs significantly worse than YYY" in a manuscript, we may want to find papers that contain similar performance evaluations (Cohan et al., 2019) and discuss these in our manuscript. This process is a non-trivial task requiring in-depth human involvement in finding, summarizing, and citing papers, which raises the question of whether it is possible to partly automate this process and make it more efficient.

Recent advances in natural language processing (NLP) help answer this question. First, releases of large scientific corpora, such as S2ORC (Lo et al., 2020) and General Index (Else, 2021), provide opportunities for building large databases of scientific papers. Second, such databases can be linked to citation recommendation (Färber and Jatowt, 2020; Gu et al., 2022b; Medić and Snajder, 2020) and to text retrieval systems (Guo et al., 2020) such as extractive summa-

rization algorithms (Zhong et al., 2020; Gidiotis and Tsoumakas, 2020; Gu et al., 2022a) and citation generation models (Xing et al., 2020; Ge et al., 2021; Wang et al., 2022) that extract sentences from scientific papers as highlights or that generate citation sentences.

These studies have focused on single tasks of either finding, reading, or summarizing. However, to build a comprehensive system that helps authors read and write scientific literature remains difficult due to the following challenges: The system needs to index a large number of papers (e.g., S2ORC has over 136 million papers (Lo et al., 2020)) to achieve good coverage, respond quickly to queries for efficient use, and be flexible to handle database additions and deletions. At the same time, the summarization algorithm needs to be efficient, and the citation generation algorithm needs to be flexible, as we want users to be able to fine-tune the generated text by modifying the keywords entered. In addition, the overall architecture needs to be modularized so one can easily upgrade each module when better algorithms become available.

To this end, we developed SciLit, a platform for joint literature discovery, summarization, and citation generation. We propose a hierarchical architecture for paper retrieval to efficiently retrieve papers from multiple large corpora. We build an efficient prefetching system based on a keyword inverted index and a document embedding index on each corpus (e.g., S2ORC and PMCOA (of Medicine, 2003)) to pre-filter candidates. The prefetched documents are then re-ordered by a fine-tuned SciBERT (Beltagy et al., 2019). Such an architecture allows us to dynamically add or remove databases and update one database and its index without affecting the others. We extract highlights using a light-weighted extractive summarization model proposed in Gu et al. (2022a) that efficiently extracts the highlights of a scientific paper by repeatedly selecting sentences in a Markov decision process. Furthermore, we fine-tune a T5 model (Raffel et al., 2020) to generate a citing sentence based on the abstract of the target paper, the context (the text surrounding the original citation sentence), and keywords provided by users. We also develop a microservice-based architecture that allows easy updates of algorithms.

In summary, our main contributions are:

- We demonstrate SciLit, a platform for one-stop searching, summarizing and citing scientific papers.
- We evaluated SciLit on scientific literature retrieval, paper summarization, and context-aware citation sentence generation, and showcased the generation of a related-work paragraph.
- A live demo website of our system is at https://scilit.vercel.app and our implementation and data are at https://github.com/nianlonggu/SciLit-React and a video demonstrating the system can be viewed at https://youtu.be/PKvNaY5Og1Y

5.2 SciLit

Figure 5.1 shows the workflow of our system. A literature discovery module receives context and keywords and recommends a list of relevant papers. For each recommended paper, an extractive summarizer selects a short list of sentences from the full text as highlights. A citation generation module takes the context, keywords, and abstract of the target paper and generates a citation sentence that references the target paper and fits the context.

We define the context as the text before a citation sentence because we focus on the workflow of first finding papers and then writing citation sentences, rather than finding the missing citation in a given sentence, like Gu et al. (2022b); Medić and Snajder (2020). In addition, the keywords are user-provided in practice. In training and evaluating our system, when no keywords are explicitly given, we use the keywords occurring in both the context, the cited paper, and the citation sentence as a substitute for use-provided keywords.

5.2.1 Literature Discovery

The literature discovery module inputs the context and keywords and recommends papers that can be cited in the current context. To strike a balance between query accuracy and speed on large scientific corpora, our document discovery module employs a two-stage prefetching-ranking strategy (Gu et al.,

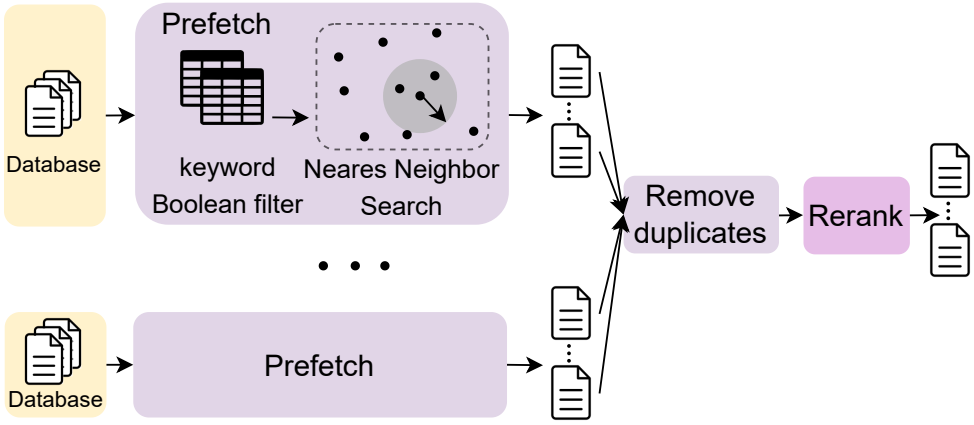


Figure 5.2: Schematic of literature retrieval. Documents of each database are prefetched by a cascade of keyword boolean filter and embedding-based nearest neighbor search and then reranked by a fine-tuned SciBERT.

Corpus	Databases		
	# of papers	# papers with fullbody	until date
S2ORC	136.60 M	12.44 M	2020-04-14
PMCOA	2.89 M	2.73 M	2022-06-17
arXiv	1.69 M	1.69 M	2022-07-28

Corpus	Inverted Index			
	keywords length	# of keywords	data format	storage size
S2ORC	unigram, bigram	1.20 B	sqlitedict	769 GB
PMCOA		0.30 B		145 GB
arXiv		0.15 B		77 GB

Corpus	Embedding Index	
	embedding dimension	storage size
S2ORC	256	169 GB
PMCOA		2.9 GB
arXiv		1.7 GB

Table 5.1: Statistics of our literature discovery system. We indexed S2ORC (Lo et al., 2020), PMCOA (of Medicine, 2003) and arXiv (Kaggle, 2022), which contain a large number of recent scientific papers in different fields.

2022b) (Figure 5.2). After importing each scientific corpus into a database, we first build an efficient prefetching model on each database to pre-filter K candidate documents based on the provided keywords and context. After removing duplicates, each database's prefetched documents are reranked to produce the final order.

Databases. We dump each corpus into a separate SQLite (Hipp, 2000) database to allow flexibility in deploying and updating prefetching servers separately. We further process documents from different corpora into a unified JSON schema so that we can use the same codebase to index, query, summarize, and display documents from different corpora. The JSON schema includes "Title", "Author", etc., for metadata, and "Content.Abstract_Parsed", "Content.Fullbody_Parsed" for parsed full text, The details are in Appendix B.

Prefetching. The prefetching model consists of an inverted index and an embedding index. The inverted index stores the paper IDs of all publications for a given keyword, such as a unigram like "computer" or a bigram like "machine learning", where the paper ID is a unique identifier with which we can obtain the paper's content from the SQLite database. The embedding index contains the embedding of each scientific paper in the database, i.e., a 256-dimensional vector representation, computed by Sent2Vec (Pagliardini et al., 2018), which computes text embeddings by simply averaging the embeddings of all words in the text. We use sentences obtained from the full text of the paper from S2ORC to train Sent2Vec.

We first perform Boolean filtering (Gökçe et al., 2020) using the inverted index based on keywords with a specific syntax. For example, given "POS tag;2010..2022", we will filter papers published between 2010 and 2022 that mention "POS tag". The filtered papers are then ranked based on the cosine similarity between the paper embedding and the context embedding computed by Sent2Vec. We believe that such a hybrid of lexical filtering and semantic ranking allows users to find papers that are semantically similar to the context and to adjust the search scope with keywords flexibly. Statistics for the database and indexing system are in Table 5.1. Details of the indexing implementation are in Appendix C.

Duplicates Removal. The prefetched candidates from multiple corpora can contain duplicated items as there exists an overlap between different corpora. To remove duplicated candidates, we check the title and authors and keep only one record of the same paper for reranking.

Reranking. We use SciBERT (Beltagy et al., 2019) to rerank prefetched candidates such that papers that the author can cite in the given context have high scores relative to the query (context and keywords). We follow Gu et al. (2022b) to compute the affinity score: An input text "[CLS]*query*[PAD]*paper*[PAD]" is passed to SciBERT, where *query* q is a concatenation of the context and keywords, and *paper* d is a concatenation of the title and abstract of the candidate paper. The encoded output of the "[CLS]" token is passed to a linear layer, which outputs a scalar $s(q, d)$ that we interpret as the affinity score between the query q and the paper d. To train the reranker we use the cross entropy loss:

$$L = -\log \frac{\exp s(q, d^+)}{\exp s(q, d^+) + \sum_{i=1}^{N} \exp s(q, d_i^-)} \tag{5.1}$$

where d^+ is the paper actually cited the query, and d_i^- is one of the $N(N = 10)$ uncited papers that are randomly sampled from prefetched candidate at each training iteration.

5.2.2 Extractive Summarization

The extractive summarization module aims to extract a short list of sentences from the full text of the paper for readers quickly get the main points. We choose the summary to be extractive rather than abstractive to prevent readers from being misled by the potential hallucinations introduced in abstractive summarization models (Nan et al., 2021; Xu et al., 2020; Wang et al., 2020). We also expect the extractive summarization model to select sentences from the entire document without truncation and summarize the document efficiently in real-time so that users do not experience obvious delays.

In this paper, we employ MemSum, an RNN-based extractive summarizer that models the extraction process as a Markov decision process in a reinforcement

learning framework. MemSum can summarize long papers without exhausting GPU memory due to its lightweight model structure, and it is computationally efficient, taking only 0.1 sec to summarize a paper. These features make it a suitable model for our extractive summarization module. We use the MemSum model trained on the PubMed dataset in Gu et al. (2022a).

5.2.3 Citation Generation Module

The citation generation module acts as an abstract summarizer that takes as input the context, keywords, and the target paper to be cited and generates a sentence that cites the target paper and narrates it in context.

Our input differs from previous work on automatic citation generation (Ge et al., 2021; Xing et al., 2020), which uses only the context as input to a sequence-to-sequence model without using keywords. We consider keywords to be an important source of input because we believe that authors usually have a clear intention when citing a paper, such as a certain keyword, rather than writing a text that summarizes the paper aimlessly. In the case shown in Figure 5.1, for example, after writing the context "MAX pooling perform worse than MEAN pooling", the author naturally intends to discuss papers about "MAX pooling". Therefore, the keyword "MAX pooling" should be used as a thematic cue for the citation sentence generation. Moreover, making the citation generation model conditional on keywords also allows users to fine-tune the generated citation text by simply adjusting the keywords, thus making our system more interactive and tunable.

To make the generation conditional on context, keywords, and cited papers, we fine-tuned a T5 (Raffel et al., 2020) so that its input is a concatenation of three input sources: keywords, context, and the abstract of a cited paper, each preceded by a special field name to make each input source distinguishable to the model: `keywords: XXX. context: XXX. target abstract: XXX.` The corresponding tag text is the actual citation sentence that cites the target paper.

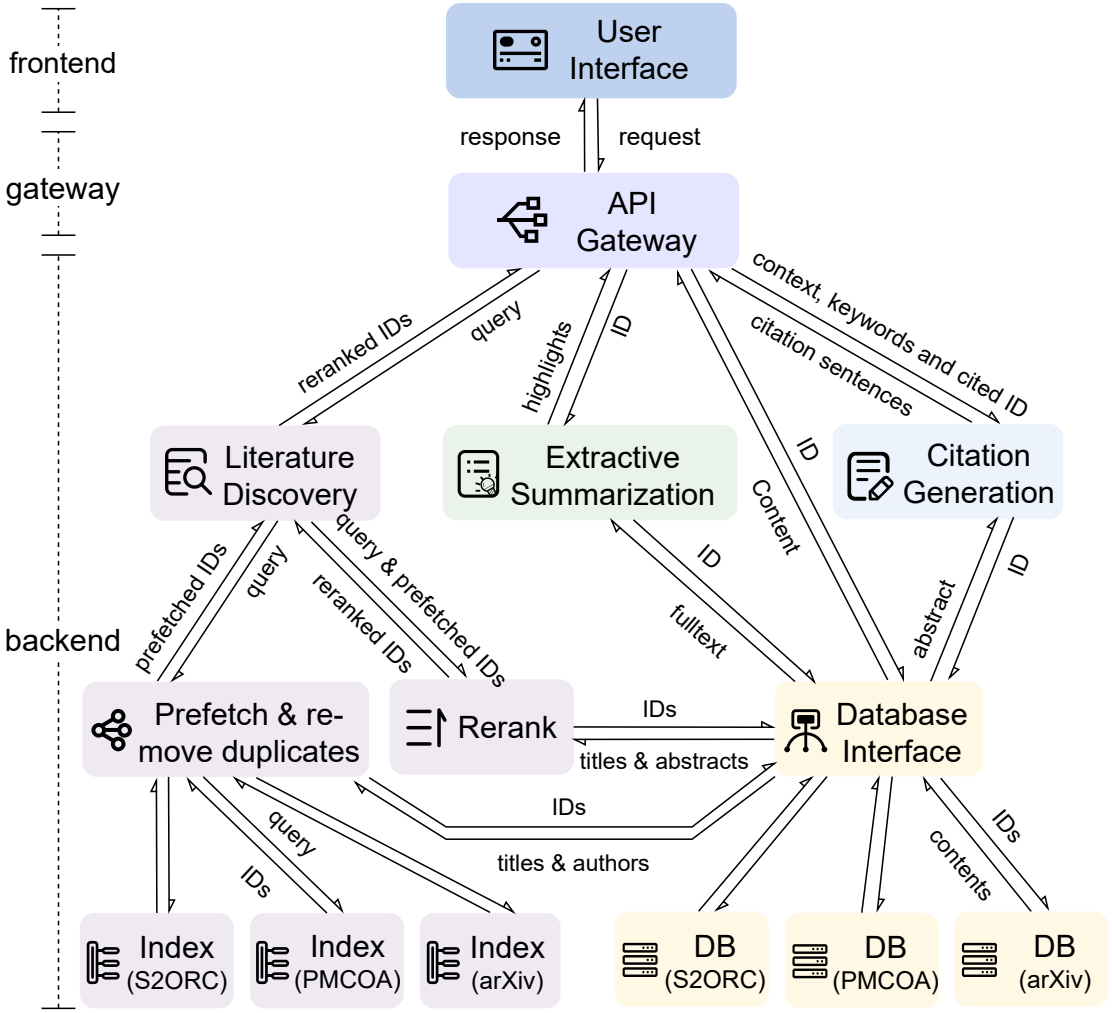


Figure 5.3: The architecture of our platform. The direction of the arrow represents the direction of data flow.

Figure 5.4: Overview of the user interface. Context text comes from the related work section in Glass et al. (2020).

5.2.4 Microservice-based Architecture

We build our platform as a network of microservices (Figure 5.3). An API gateway routes requests from the frontend to the target microservice on the backend. The microservices run separate modules on their respective Flask servers (Aggarwal, 2014) and communicate with each other by sending HTTP requests and waiting for responses. When a query request arrives, the API gateway forwards the query to the literature discovery service, which calls the prefetching and reranking service to get the reranked IDs. The API gateway then sends the paper IDs to the extractive summarization service for the highlights of each recommended paper and sends the context, keywords, and recommended paper IDs to the citation generation service to suggest citation sentences. The database interface service manages the databases of multiple scientific corpora and provides a unified interface to access the paper content for a given ID. Each microservice runs in a relatively independent environment that can be viewed externally as a black box, which makes it easy to upgrade backend systems online, such as adding or removing a database or updating a module's algorithm.

5.3 Evaluation

In this section we first show how SciLit works and then evaluate the performance of the system.

5.3.1 Demonstration

Our user interface is set up on a web page (Figure 5.4), implemented in ReactJS[1]. The left sidebar is an input panel where users can enter context and keywords and start a query by clicking the search button. Retrieved papers' information is displayed in the search results panel on the right. Users can scroll up and down or paginate to browse through the recommended papers. Each paper is accompanied by highlights and a suggested citation sentence

[1] https://reactjs.org/

N_p	speed (s/query)	Recall@K (R@K)					
		R@1	R@5	R@10	R@20	R@50	R@100
50	**2.02**	**0.107**	0.208	0.263	0.305	0.327	0.331
100	2.55	0.096	0.215	**0.278**	0.328	0.384	0.401
200	3.26	0.095	**0.220**	0.275	**0.339**	0.420	0.452
300	3.93	0.095	0.204	0.273	0.330	**0.422**	**0.482**

Table 5.2: Literature retrieval performance measured by the recall of top K recommendations. N_p denotes the number of prefetched candidates per corpus.

generated by our extractive summarization service and citation generation service, respectively. Users can cite a paper by clicking on the cite button and the suggested citation sentence will jump to the editing area on the left where users can tweak the sentence by changing keywords and clicking on the fine-tune generation button, or they can edit the sentences manually. Exporting citation information is also supported.

5.3.2 Performance

Evaluation Dataset. We evaluated SciLit on a test set containing 1530 samples, mainly from papers published in 2022 in the fields of computer science and biomedical science. Each sample contains the following information: 1) context, up to 6 sentences that precede the citation sentence and are within the same section as the citation sentence; 2) keywords, up to 2 uni- or bi-grams that occur in both the context, the citation sentence and cited paper; 3) ID of the cited paper; 4) the citation sentence following the context, which is the ground truth when evaluating generated citations. We only include citation sentences that cite one paper in the test set for better quality control.

Literature Retrieval. For each sample in the evaluation dataset, we use context and keywords as queries and invoke the literature search service to first prefetch N_p candidates from three corpora (S2ORC, PMCOA, and arXiv) respectively, remove duplicates, and then rank the prefetched candidates to obtain the recommended papers. We evaluated the retrieval performance based on the recall of the real cited papers in the top K recommendations (Table

Model	Rouge-1	Rouge-2	Rouge-L
BertSum (Liu, 2019)	42.53	16.89	39.18
MemSum (Gu et al., 2022a)	**46.40***	**19.61***	**42.66***

Table 5.3: The extractive summarization performance. "*" indicates statistical significance in comparison to baselines with a 95% bootstrap confidence interval.

generation pipeline	Rouge-1	Rouge-2	Rouge-L
generation-only	32.96	9.19	24.52
Best of top 1 paper	28.62	6.00	21.05
Best of top 5 papers	34.92	9.59	26.23
Best of top 10 papers	**36.83***	**10.98***	**28.10***

Table 5.4: The performance of citation generation.

5.2). We observed that for large $K(K = 50, 100)$, the recall increases as N_p increases, while for small $K(K = 5, 10, 20)$, the recall first increases and then starts to decrease, indicating that the reranking performance is impacted by more prefetched candidates. We choose 100 as the default value for N_p, which achieved the best performance on R@10 and has a relatively fast query speed.

Extractive Summarization. Following Zhong et al. (2020); Xiao and Carenini (2019), we computed the ROUGE f1 scores between the highlights extracted from the full body and the corresponding abstract. MemSum significantly outperformed BertSum (Liu, 2019), a Bert-based summarizer that requires truncation before summarizing long documents, indicating the effectiveness of MemSum in extractively summarizing scientific documents.

Citation Generation. To evaluate our joint retrieval and citation generation pipeline, we have our system first recommend papers based on context and keywords and then generate K citation sentences for each of the top K recommended papers, respectively. Then, we calculate the ROUGE scores between the real citation and the K generated sentences and record the highest of them. We compared the "Best-of-top-K" pipeline to the "generation-only" pipeline, where we directly provide the truly cited papers for generation citations.

retrieval ($N_p = 100$)	R@1	R@5	R@10	R@20	R@50	R@100
w keywords	0.096	0.215	0.278	0.328	0.384	0.401
w/o keywords	0.013	0.050	0.085	0.125	0.199	0.250

citation generation	Rouge-1	Rouge-2	Rouge-L
w keywords	32.96	9.19	24.52
w/o keywords	26.57	5.56	20.39

Table 5.5: Ablation study on retrieval and citation generation performance.

We observed that for $K = 5$ and 10, the "Best-of-top-K" pipeline achieved significantly higher ROUGE scores than the "generation only" pipeline (Table 5.4), indicating that the paper retrieval module contributes positively to the citation generation process, with a greater chance of suggesting more appropriate citation sentences. We believe that this result further supports our idea of developing an integrated system for joint retrieval and generation.

5.3.3 Ablation Study

To analyze the impact of keywords, we evaluated retrieval and generation systems without keywords. For document retrieval, we first prefetch $N_p = 100$ candidates from each corpus and then rank them based on context only. For citation generation, we trained a T5 model to learn to generate citation sentences with only the context and the title and abstract of the cited paper and evaluated it on the evaluation dataset. We observe a significant degradation in the performance of literature retrieval and citation generation (Table 5.5), which demonstrates the necessity of keywords for recommending relevant papers and generating accurate citations when using our platform.

5.4 Related Work

Recently, AI-driven platforms focused on literature recommendation and scientific paper summarization have been proposed, respectively. (keywords: `platform`, paper: #2) *One such platform is AI Research Navigator (Fadaee et al.,*

2020), which combines classical keyword search with neural retrieval to discover and organize relevant literature. (keywords: `scientific; summarization; platform`, paper #3) *Another platform is Anne O'Tate, which supports user-driven summarization, drill-down and mining of search results from PubMed, the leading search engine for biomedical literature (Smalheiser et al., 2021).* (keywords: `related work generation`, paper #9) *Chen and Zhuge (2019) automatically generates related work by comparing the main text of the paper being written with the citations of other papers that cite the same references.*

In the previous paragraph, the italicized citation sentences are generated from SciLit. In generating each sentence, we use all the preceding sentences in the paragraph as contexts and use the keywords in parentheses to obtain the recommended papers and the corresponding citation sentences. The paper index in parentheses indicates the order of the final selected cited papers in the recommended results.

5.5 Conclusion and Future Work

This paper demonstrates SciLit, a platform for joint scientific literature retrieval, paper summarization, and citation generation. SciLit can efficiently recommend papers from hundreds of millions of papers and proactively provide highlights and suggested citations to assist authors in reading and discussing relevant papers. In addition, we have our prefetching, reranking, and citation generation system conditioned on user-provided keywords, which makes our platform more flexible and adjustable in response to user input when recommending papers and suggesting citation sentences. In the future, we will further improve the performance of each module, especially the citation generation part, and collect feedback from users to improve the overall workflow and the frontend user experience.

5.6 Appendices

A Hardware Information

We run the backend of SciLit on a server with 2×64 Core AMD EPYC 7742 2.25GHz Processor, 2TB DDR4 3200MHz ECC Server Memory, and 4×7.68TB NVME GEN4 PM9A3 for storage. The server is also equipped with 2 × nVidia RTX A6000 48GB GPU. The frontend is hosted on Vercel[2].

B JSON Schema for Database

The details of our unified JSON schema is shown in Listing 5.1. We keep the fields: "Author", "Title", "Abstract", "Venue", "DOI", "URL" and "PublicationDate" for the metadata, and "Content.Abstract_Parsed" and "Content.Fullbody_Parsed" for the parsed full text. The parsed abstract or full body contains a list of parsed sections. Each section contains a list of parsed paragraphs, each including a list of parsed sentences. If a sentence cites a paper, we create a "cite span" that records the citation marker such as "[1]", the position of the citation marker in the sentence, and the cited paper's index in the "Reference" list.

We implemented an S2ORC parser to convert documents in the S2ORC corpus to our JSON format. For PDFs in the arXiv corpus, we first used the s2orc-doc2json (Lo et al., 2020) to convert them into S2ORC format and then applied our S2ORC parser. For XML files in the PMCOA corpus, we implemented an XML parser based on Achakulvisut et al. (2020) to convert XML to S2ORC format and then applied the S2ORC parser to convert it into our JSON format finally.

C Prefetching Indexing Implementation

Inverted Index

The inverted index is a mapping table from keywords (unigrams or bigrams) to paper IDs. We extract keywords from the full text of each document and

[2] https://vercel.com/

```
{'Author': [{'GivenName': 'Daisuke', 'FamilyName': 'Ida'}, ...],
 'Title': 'Topology Change of Black Holes',
 'Abstract': 'The topological structure of the event horizon has
   been investigated ...',
 'Venue': '',
 'DOI': '',
 'URL': '',
 'PublicationDate': {'Year': '2007', 'Month': '3'},
  'Content': {
     'Abstract': '',
     'Abstract_Parsed': [{
         'section_id': '0',
         'section_title': 'Abstract',
         'section_text': [{
             'paragraph_id': '0',
             'paragraph_text': [{
                 'sentence_id': '0',
                 'sentence_text': 'The topological structure of
   the event horizon has been investigated in terms of the Morse
   theory.',
                 'cite_spans': []},
                 # ...
             ]},
             # ...
         ]
     }],
     'Fullbody': '',
     'Fullbody_Parsed': [{
         'section_id': '0',
         'section_title': 'Introduction',
         'section_text': [{
             'paragraph_id': '0',
             'paragraph_text': [
                 # ...,
                 {
                 'sentence_id': '2',
                 'sentence_text': '[1, 2] This follows from the
   fact that the total curvature, which is the integral of the
   intrinsic scalar curvature over the horizon, is positive under
    the dominant energy condition and from the Gauss-Bonnet
   theorem.',
                 'cite_spans': [{'start': '4', 'end': '6', 'text':
    '2]','ref_id': '0'}]
                 },
                 # ...
             ]
         }]
     }]},
 'Reference': [{
     'Title': 'The large scale structure of space-times',
     'Author': [{'GivenName': 'S', 'FamilyName': 'Hawking'},
                {'GivenName': 'G', 'FamilyName': 'Ellis'}],
     'Venue': '',
     'PublicationDate': {'Year': '1973'},
     'ReferenceText': '2. Hawking, S, and G Ellis. "The large
   scale structure of space-times." (1973).'},
     # ...
     ]
}
```

```
{
    'operation': 'AND',
    'elements': [
        {'operation': 'AND',
         'elements': [{'operation': None, 'elements': ['nlp']}]},
        {'operation': 'OR',
         'elements': [
            {'operation': 'AND',
             'elements': [{'operation': None,
                           'elements': ['machine translation']}]},
            {'operation': 'AND',
             'elements': [{'operation': None, 'elements': ['nmt'
    ]}]}]},
        {'operation': 'OR',
         'elements': [
            {'operation': None, 'elements': ['publicationdate.year
    :2020']},
            {'operation': None, 'elements': ['publicationdate.year
    :2021']},
            {'operation': None, 'elements': ['publicationdate.year
    :2022']}
         ]
        }]
}
```

Listing 5.2: The dictionary representation of the tree structure shown in Figure 5.5.

keep a bigram only if all two words in the bigram are not English stopwords. We use sqlitedict[3] to store the inverted index for each corpus, which is an on-disk hashmap based on an SQLite database that allows us to efficiently obtain the paper ID for a given keyword without loading the entire inverted index into RAM.

Syntax Parsing. Our platform allows users to filter documents using syntax-rich keyword strings. For example, to filter papers that contain the keywords 'NLP' and 'machine translation' or 'NMT' published between 2020 and 2022, one can compose a keyword string `NLP; machine learning|NMT; 2020..2022''`. We transform this keyword string into a tree of logical operations (Figure 5.5), wherein each node we denote the logical operations applied to the sub-nodes, and each leaf node contains a keyword. We implemented the tree using a

[3] https://github.com/RaRe-Technologies/sqlitedict

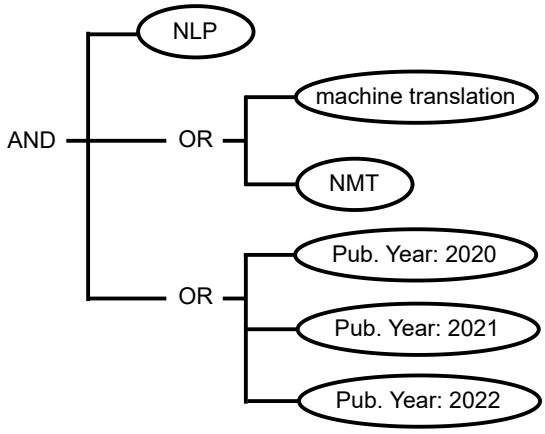


Figure 5.5: The parsed tree structure of the given keywords string: `NLP; machine learning|NMT; 2020..2022''`.

Python dictionary (Listing 5.2). Then, we recursively traverse all nodes in the tree in a depth-first search, obtain the paper IDs with the keyword in each leaf node, and apply the logical operations indicated in each node to obtain the final paper ID at the root node.

Embedding Index

Structure of the Embedding Index. The embedding index consists of three main components:

The first component is a matrix $\mathcal{M} \in \mathcal{R}^{N \times D}$, where N is the number of documents and D is the dimensionality of document embeddings. Each document's embedding is L2-normalized so that given an L2-normalized query embedding $e_q \in \mathcal{R}^{D \times 1}$, the matrix multiplication $\mathcal{M}e_q \in \mathcal{R}^{N \times 1}$ represents the cosine similarity between the query embedding and all document embeddings. We can use $\mathcal{M}e_q$ to rank documents to obtain the indices of most similar paper embeddings.

The second component is a mapping table from the index of a paper embedding in the matrix $\mathcal{M}$ to the corresponding paper ID in our databases. With this mapping table we can get the papers' content given the top ranked indices during K nearest neighbor search (KNN).

The last component is a reversed mapping table from the paper ID to the corresponding index in the embedding matrix. In our prefetching system, we first use the inverted index to pre-filter a subset of paper IDs based on given keywords. Then we can use this reversed mapping table to obtain the corresponding paper embeddings and perform KNN among them.

Multi-Processing Speedup for Brute-Force Nearest Neighbor Search. For a large corpus like S2ORC, the embedding matrix contains up to 136.6 million vectors, and performing matrix multiplication in a single thread is very time-consuming. To take full advantage of the multiple CPU cores on our server, we divide the embedding matrix into 137 shards, each containing about 1 million embeddings. We first run a brute-force nearest neighbor search in parallel to obtain N_p candidates on each shard, and then we rank the $137 \times N_p$ candidates again to obtain the final N_p candidates. Given that our server has 128 cores, we can achieve a nearly linear speedup using multiprocessing KNN with slicing, and mathematically it is equivalent to performing a single KNN over the entire embedding matrix to obtain the closest N_p candidates.

D Joint Retrieval and Citation Generation Examples

We show some specific results of joint paper retrieval and automatic generation of citation sentences. The contexts and keywords we used were obtained from papers in arXiv (Figure 5.6) and PMCOA (Figure 5.7), respectively. In each example, the actual cited paper occurs in the top 5 paper recommendations, which we have highlighted with an underline.

Context:

#OTHERCIT apply a multi-stream CNN model to extract and fuse deep features from the designed complementary shape-mot ion representations. Zhu et al. #OTHERCIT organize the pairwise displacements between all body joints to obtain a cu boid action representation and use attention-based deep CNN models to focus analysis on actions. Inspired by the fac t that the skeleton data is naturally a topological graph, where the joints and bones are regarded as the nodes and edges, Graph Convolutional Network (GCN) is adopted to boost the performance of skeleton based action recognition #O THERCITS .

Keywords:

skeleton

Recommended Papers and Generated Citations:

UNIK: A Unified Framework for Real-world Skeleton-based Action Recognition
Generated Citation: UNIK #CIT is a generic skeleton-based action recognition model pre-trained on Posetics, a large -scale human skeleton video dataset.

ANUBIS: Skeleton Action Recognition Dataset, Review, and Benchmark
Generated Citation: The ANUBIS dataset #CIT is a large-scale 3D skeleton dataset, which mainly consists of 3D joint s, bones, and limbs.

SkeletonNet: Mining Deep Part Features for 3-D Action Recognition
Generated Citation: SkeletonNet #CIT is proposed to extract body-part-based features from each frame of the skeleto n sequence, which are translation, rotation, and scale invariant.

Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition
Generated Citation: ST-GCN #CIT learns both the spatial and temporal patterns from the skeleton data, which leads t o greater expressive power and stronger generalization capability.
Real Citation: ST-GCN #CIT firstly constructs the skeleton as a graph and uses GCN to automatically capture the pat terns embedded in the spatial configuration of the joints as well as their temporal dynamics.

SKELETON-BASED ACTION RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS
Generated Citation: In #CIT, a novel skeleton transformer module is designed to rearrange and select important skel eton joints automatically.

Figure 5.6: Example of joint paper retrieval and citation generation. The context text was obtained from arXiv.

Context:

Quadricuspid aortic valve (QAV) is a rare congenital heart disease that is less common than bicuspid and unicuspid aortic valve [#OTHERCITS]. Coronary artery and coronary ostium anomalies are present in 2–10% of patients with QAV [#OTHERCIT, #OTHERCIT, #OTHERCITS]. The functional status of QAV is predominantly pure aortic regurgitation (AR) [#OTHERCIT, #OTHERCIT].

Keywords:

functional status

Recommended Papers and Generated Citations:

ESC Guidelines for the management of grown-up congenital heart disease (new version 2010): The Task Force on the Management of Grown-up Congenital Heart Disease of the European Society of Cardiology (ESC)
Generated Citation: The functional status of QAV is characterized by the presence of aortic regurgitation (AR), aortic stenosis (AR), and aortic stenosis (AR) [#CIT].

Quadricuspid Aortic Valve: A Comprehensive Review
Generated Citation: The clinical manifestations of QAV depend on the functional status of the QAV and the associated disorders [#CIT].
Real Citation: Clinical manifestations, such as palpitations, dyspnea, fatigue, and chest pain, depend on the functional status of QAV and usually present in the fifth or sixth decade of life [#CIT].

A classification system for the bicuspid aortic valve from 304 surgical specimens.
Generated Citation: The functional status of QAV is characterized by the presence of a number of raphes, spatial position of cusps or raphes, and number of raphes [#CIT].

Functional status of the quadricuspid aortic valve/an uncommon coincidence of congenital quadricuspid aortic valve accompanied by hypertrophic obstructive cardiomyopathy.
Generated Citation: The functional status of QAV is characterized by hypertrophic obstructive cardiomyopathy (HOMC) and hypertrophic obstructive cardiomyopathy (HOCM) [#CIT].

Congenital coronary artery fistulae: a rare cause of heart failure in adults
Generated Citation: The functional status of QAV can be improved by epicardial fistula ligation and coronary artery bypass grafting with marked improvement in functional status [#CIT].

Figure 5.7: Example of joint paper retrieval and citation generation. The context text was obtained from PMCOA.

Chapter 6

Conclusion and Future Work

This thesis investigated using NLP techniques to assist authors in finding, summarizing, and citing papers while writing scientific articles. We explored three separate research topics: citation recommendation, long document extractive summarization, and controlled citation generation, and finally, we integrated our algorithms into a holistic system and built a web platform that allows users to use our workflow efficiently.

6.1 Paper Retrieval

In Chapter 2, we proposed the HAtten-SciBERT recommendation pipeline, which consists of a prefetching phase and a reranking phase. We first rank the papers in the database based on the cosine similarity between the query embedding and the paper embedding encoded by HAtten, a hierarchical attentional text encoder. We then rerank the prefetched candidate papers using fine-tuned SciBERT. Our HAtten model efficiently prefetches a list of candidates with significantly higher recall than baselines, which allows SciBERT to rerank a smaller number of prefetched candidates in less time while maintaining recall performance, thus achieving a better tradeoff between efficiency and accuracy. Our results show that not only reranking performance but also prefetching performance is critical to the effectiveness of the whole pipeline and must be considered when designing practical citation recommendation

pipelines.

One limit in our approach is that the training of the HAtten text encoder converges slowly. We train HAtten based on triplet loss. For each query text, there is usually only one positive document, i.e., the actual cited paper, and millions of negative documents (uncited papers). The severe imbalance between positive and negative documents makes it crucial to mine effective triplet pairs. We investigated triplet mining strategies to speedup convergence, but there is still much room for improvement. Future work could explore more effective triplet mining strategies or replace triplet loss with other metric learning losses (KAYA and BİLGE, 2019) to enable fast training convergence, especially in large databases with millions of papers. Furthermore, we observe that the reranking time is roughly linearly proportional to the number of papers to be reordered, suggesting that the reranking part will become a time bottleneck when we want to rerank more prefetched candidates in pursuit of higher recall. Future work could investigate more efficient reranking systems than SciBERT, such as using lightweight variants of BERT like ALBERT (Lan et al., 2019) or DistilBERT (Sanh et al., 2019).

6.2 Document Summarization

In Chapter 3. we modeled extractive summarization as a multi-step episodic Markov decision process with extractive history awareness. In each step, we score the remaining unextracted sentences based not only on the content of the sentences but also on extraction history information, i.e., which sentences have already been extracted and which are remaining. The awareness of extraction history allows MemSum to produce compact summaries and to avoid bringing redundancy in the extracted summary spontaneously. In addition, the lightweight structure gives MemSum an advantage in summarizing long documents, which makes it suitable for summarizing scientific papers.

One limit of recent extractive summary models (including this work) is the lack of consideration of coherence (i.e., the logical connection between sentences in an extractive summary), which is evidenced by the fact that the popular auto-

matic assessment metrics, ROUGE-1/2/Lsum F1 scores, are insensitive to the order of extracted sentences. We believe that coherence is an essential factor, as extracted sentences ordered in a logically consistent way certainly help the reader better grasp the original document's logical flow. Future work could investigate the development of evaluation metrics for coherence. With such metrics, we can design new reward functions and train MemSum through the policy gradient approach to extract sentences coherently. Moreover, developing an extractive-and-abstractive pipeline remains an interesting topic where we first extract a subset of sentences from a document and then rephrase them with a fine-tuned language model.

6.3 Controllable Citation Generation

In Chapter 4, we presented a controlled citation generation system that allows users to use suggested citation attributes (intent, keywords, or related sentences) to guide generation. The key module of the system is a BART-large model trained using not only contextual inputs but also conditional inputs. The contextual inputs consist of the local context in the manuscript and the cited paper's global context (title and abstract). The conditional inputs include the citation attributes associated with the target citation sentence. The experimental results showed that training BART-large with conditional inputs could facilitate generation conditional on the given citation attributes, thus allowing the user to control the generation process effectively.

Future work could investigate ways to improve the controllability of citation generators. For example, researchers could refine the definition of citation attributes, such as defining more types of citation intent instead of only three (context, method, result). On the other hand, the controllability of citation generation models can also be achieved by tuning the decoding process via updating the probability of the next token with a given control property, as proposed in Dathathri et al. (2020).

6.4 SciLit

In Chapter 5, we presented a workflow for automatically recommending papers, providing an extracted summary as highlights for each paper, and suggesting citation sentences that can be used to cite the recommended papers. Users only need to provide the manuscript they are working on and some keywords they would like to discuss next. The backbone of this platform is based on our work on paper retrieval (Chapter 2), summarization (Chapter 3), and citation generation (Chapter 4).

Future work can focus on the following aspects. First, we can further improve the efficiency of our paper retrieval system in recommending papers. Our current platform takes around 2.6 s to recommend papers from a database of around 140 million (by prefetching 100 candidates and then reranking them), and we aim to reduce the time to less than 1 s. Second, we will conduct a user study on the platform and collect user feedback to improve human-machine interaction and the overall workflow.

6.5 Overall Conclusion

In this thesis, we investigated the NLP-driven task of finding, summarizing, and citing relevant papers in scientific writing. Citation generation is a process of making a statement based on two input sources: 1) the contextual information in the manuscript, which may contain observations (e.g., experimental results), and 2) information about the cited paper, which is a priori knowledge. Thus, the citation generation task can be considered as a specific case of scientific inference on a microscopic scale. On a broader scale, given that writing the discussion section usually involves citing and discussing the relevant papers, solving the problem of citation generation will facilitate the process of writing the discussion section. Thus, our work is a step towards fully automated discussion generation, which helps us to implement AI-driven scientific inference at the section (or chapter) scale.

Bibliography

Titipat Achakulvisut, Daniel Acuna, and Konrad Kording. Pubmed parser: A python parser for pubmed open-access xml subset and medline xml dataset xml dataset. *Journal of Open Source Software*, 5(46):1979, 2020. doi: 10.21105/joss.01979. URL https://doi.org/10.21105/joss.01979.

Shalabh Aggarwal. *Flask framework cookbook*. Packt Publishing Ltd, 2014.

Zafar Ali, Guilin Qi, Khan Muhammad, Pavlos Kefalas, and Shah Khusro. Global citation recommendation employing generative adversarial network. *Expert Systems with Applications*, 180:114888, 2021a. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.114888. URL https://www.sciencedirect.com/science/article/pii/S0957417421003298.

Zafar Ali, Guilin Qi, Khan Muhammad, Asim Khalil, Inam Ullah, and Amin Khan. Global citation recommendation employing multi-view heterogeneous network embedding. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2021b. doi: 10.1109/CISS50987.2021.9400311.

Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/

v1/D19-1371. URL https://aclanthology.org/D19-1371.

Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. Content-based citation recommendation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 238–251, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1022. URL https://www.aclweb.org/anthology/N18-1022.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336, 1998.

Jingqiang Chen and Hai Zhuge. Automatic generation of related work through summarizing citations. *Concurrency and Computation: Practice and Experience*, 31(3):e4261, 2019. ISSN 1532-0634. doi: 10.1002/cpe.4261. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4261. Number: 3 _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4261.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2097. URL https://www.aclweb.org/anthology/N18-2097.

Arman Cohan, Waleed Ammar, Madeleine van Zuylen, and Field Cady. Structural scaffolds for citation intent classification in scientific publications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3586–3596, Minneapolis, Minnesota, June 2019.

Association for Computational Linguistics. doi: 10.18653/v1/N19-1361. URL https://aclanthology.org/N19-1361.

Jacob Cohen. *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data, 2017. URL https://arxiv.org/abs/1705.02364.

Tao Dai, Li Zhu, Yaxiong Wang, and Kathleen M. Carley. Attentive stacked denoising autoencoder with bi-lstm for personalized context-aware citation recommendation. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 28:553–568, January 2020. ISSN 2329-9290. doi: 10.1109/TASLP.2019.2949925. URL https://doi.org/10.1109/TASLP.2019.2949925.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. *arXiv:1912.02164 [cs]*, March 2020. URL http://arxiv.org/abs/1912.02164. arXiv: 1912.02164.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv: 1810.04805.

Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi Kit Cheung. BanditSum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3739–3748, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1409. URL https://www.aclweb.org/anthology/D18-1409.

Travis Ebesu and Yi Fang. Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17,

page 1093–1096, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450350228. doi: 10.1145/3077136.3080730. URL https://doi.org/10.1145/3077136.3080730.

Vladimir Eidelman. BillSum: A corpus for automatic summarization of US legislation. In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, 2019. doi: 10.18653/v1/d19-5406. URL https://doi.org/10.18653%2Fv1%2Fd19-5406.

Holly Else. Giant, free index to world's research papers released online. available online at https://www. nature. com/articles/d41586-021-02895-8, accessed 2021-12-15. *Nature (Oct 2021). https://doi. org/10.1038/d41586-021-02895-8*, 2021.

Marzieh Fadaee, Olga Gureenkova, Fernando Rejon Barrera, Carsten Schnober, Wouter Weerkamp, and Jakub Zavrel. A new neural search and insights platform for navigating and organizing AI research. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 207–213, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sdp-1.23. URL https://aclanthology.org/2020.sdp-1.23.

Michael Färber and Ashwath Sampath. Hybridcite: A hybrid model for context-aware citation recommendation. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, JCDL '20, page 117–126, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375856. doi: 10.1145/3383583.3398534. URL https://doi.org/10.1145/3383583.3398534.

Michael Färber, Timo Klein, and Joan Sigloch. Neural citation recommendation: A reproducibility study. In *BIR@ECIR*, 2020.

Jessica Ficler and Yoav Goldberg. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4912. URL https://aclanthology.org/W17-4912.

Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3207. URL https://www.aclweb.org/anthology/W17-3207.

Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proc. VLDB Endow.*, 12(5):461–474, January 2019. ISSN 2150-8097. doi: 10.14778/3303753.3303754. URL https://doi.org/10.14778/3303753.3303754.

Michael Färber and Adam Jatowt. Citation recommendation: approaches and datasets. *International Journal on Digital Libraries*, 21(4):375–405, December 2020. ISSN 1432-5012, 1432-1300. doi: 10.1007/s00799-020-00288-2. URL http://link.springer.com/10.1007/s00799-020-00288-2. Number: 4.

Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: A survey. *Artif. Intell. Rev.*, 47(1):1–66, January 2017. ISSN 0269-2821. doi: 10.1007/s10462-016-9475-9. URL https://doi.org/10.1007/s10462-016-9475-9.

Yubin Ge, Ly Dinh, Xiaofeng Liu, Jinsong Su, Ziyao Lu, Ante Wang, and Jana Diesner. BACO: A Background Knowledge- and Content-Based Framework for Citing Sentence Generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1466–1478, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.116. URL https://aclanthology.org/2021.acl-long.116.

Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040, 2020. doi: 10.1109/TASLP.2020.3037401.

Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G P Shrivatsa Bhargav, Dinesh Garg, and Avi Sil. Span selection pre-training

for question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2782, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.247. URL https://aclanthology.org/2020.acl-main.247.

Onur Gökçe, Jonathan Prada, Nikola I. Nikolov, Nianlong Gu, and Richard H.R. Hahnloser. Embedding-based scientific literature discovery in a text editor application. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 320–326, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.36. URL https://aclanthology.org/2020.acl-demos.36.

Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020. URL https://doi.org/10.5281/zenodo.4461265.

Nianlong Gu, Elliott Ash, and Richard Hahnloser. MemSum: Extractive summarization of long documents using multi-step episodic Markov decision processes. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6507–6522, Dublin, Ireland, May 2022a. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.450. URL https://aclanthology.org/2022.acl-long.450.

Nianlong Gu, Yingqiang Gao, and Richard H. R. Hahnloser. Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking. In Matthias Hagen, Suzan Verberne, Craig Macdonald, Christin Seifert, Krisztian Balog, Kjetil Nørvåg, and Vinay Setty, editors, *Advances in Information Retrieval*, pages 274–288, Cham, 2022b. Springer International Publishing. ISBN 978-3-030-99736-6.

Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 57(6): 102067, 2020.

Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and Lee Giles. Context-aware

citation recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 421–430, 2010.

Gustav Herdan. *Type-token mathematics*, volume 4. Mouton, 1960.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching Machines to Read and Comprehend. *arXiv:1506.03340 [cs]*, November 2015. URL http://arxiv.org/abs/1506.03340. arXiv: 1506.03340.

Richard D Hipp. Sqlite. https://www.sqlite.org/index.html, 2000. Version: 3.39.2, Accesses: 2022-07-31.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.112. URL https://aclanthology.org/2021.naacl-main.112.

Wenyi Huang, Saurabh Kataria, Cornelia Caragea, Prasenjit Mitra, C Lee Giles, and Lior Rokach. Recommending citations: translating papers into references. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1910–1914, 2012.

Lawrence Hunter and K Bretonnel Cohen. Biomedical language processing: what's beyond pubmed? *Molecular cell*, 21(5):589–594, 2006.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

Chanwoo Jeong, Sion Jang, Eunjeong L. Park, and Sungchul Choi. A context-aware citation recommendation model with BERT and graph con-

volutional networks. *Scientometrics*, 124(3):1907–1922, 2020. doi: 10.1007/s11192-020-03561-y. URL https://doi.org/10.1007/s11192-020-03561-y.

Ruipeng Jia, Yanan Cao, Hengzhu Tang, Fang Fang, Cong Cao, and Shi Wang. Neural extractive summarization with hierarchical attentive heterogeneous graph network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3622–3631, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.295. URL https://aclanthology.org/2020.emnlp-main.295.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

Shing-Yun Jung, Ting-Han Lin, Chia-Hung Liao, Shyan-Ming Yuan, and Chuen-Tsai Sun. Intent-controllable citation text generation. *Mathematics*, 10(10):1763, 2022.

David Jurgens, Srijan Kumar, Raine Hoover, Dan McFarland, and Dan Jurafsky. Measuring the evolution of a scientific field through citation frames. *Transactions of the Association for Computational Linguistics*, 6:391–406, 2018. doi: 10.1162/tacl_a_00028. URL https://aclanthology.org/Q18-1028.

Kaggle. arXiv Dataset. https://www.kaggle.com/datasets/Cornell-University/arxiv, 2022. Accesses: 2022-08-01.

Mahmut KAYA and Hasan Şakir BİLGE. Deep metric learning: A survey. *Symmetry*, 11(9), 2019. ISSN 2073-8994. doi: 10.3390/sym11091066. URL https://www.mdpi.com/2073-8994/11/9/1066.

Lord Kelvin. I. nineteenth century clouds over the dynamical theory of heat and light. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(7):1–40, 1901.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019. URL https://arxiv.org/abs/1909.05858.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference*

on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. URL http://arxiv.org/abs/1412.6980.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Yuta Kobayashi, Masashi Shimbo, and Yuji Matsumoto. Citation recommendation using distributed representation of discourse facets in scientific articles. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, JCDL '18, page 243–251, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450351782. doi: 10.1145/3197026.3197059. URL https://doi.org/10.1145/3197026.3197059.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2019. URL https://arxiv.org/abs/1909.11942.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://www.aclweb.org/anthology/2020.acl-main.703.

Xiangci Li and Jessica Ouyang. Automatic related work generation: A meta study, 2022. URL https://arxiv.org/abs/2201.01880.

Pengcheng Liao, Chuang Zhang, Xiaojun Chen, and Xiaofei Zhou. Improving

abstractive text summarization with history aggregation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE, 2020.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Peter J. Liu*, Mohammad Saleh*, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Hyg0vbWC-.

Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction, 2019. URL https://arxiv.org/abs/1901.05567.

Yang Liu. Fine-tune bert for extractive summarization, 2019. URL https://arxiv.org/abs/1903.10318.

Yang Liu and Mirella Lapata. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy, July 2019a. Association for Computational Linguistics. doi: 10.18653/v1/P19-1500. URL https://www.aclweb.org/anthology/P19-1500.

Yang Liu and Mirella Lapata. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China, November 2019b. Association for Computational Linguistics. doi: 10.18653/v1/D19-1387. URL https://www.aclweb.org/anthology/D19-1387.

Avishay Livne, Vivek Gokuladas, Jaime Teevan, Susan T. Dumais, and Eytan Adar. Citesight: Supporting contextual citation recommendation using differential search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, page

807–816, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450322577. doi: 10.1145/2600428.2609585. URL https://doi.org/10.1145/2600428.2609585.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.447. URL https://www.aclweb.org/anthology/2020.acl-main.447.

Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang, and Qing He. Reading like her: Human reading inspired extractive summarization. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3024–3034, 2019.

Mani Malek Esmaeili, Rabab Kreidieh Ward, and Mehrdad Fatourechi. A Fast Approximate Nearest Neighbor Search Algorithm in the Hamming Space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2481–2488, December 2012. ISSN 1939-3539. doi: 10.1109/TPAMI.2012.170. Number: 12.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008. ISBN 978-0-521-86571-5. URL http://nlp.stanford.edu/IR-book/information-retrieval-book.html.

Zoran Medić and Jan Snajder. Improved Local Citation Recommendation Based on Context Enhanced with Global Information. In *Proceedings of the First Workshop on Scholarly Document Processing*, pages 97–103, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sdp-1.11. URL https://www.aclweb.org/anthology/2020.sdp-1.11.

Farida Mohsen, Jiayang Wang, and Kamal Al-Sabahi. A hierarchical self-

attentive neural extractive summarizer via reinforcement learning (hsasrl). *Applied Intelligence*, pages 1–14, 2020.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 3075–3081. AAAI Press, 2017.

Ramesh M Nallapati, Amr Ahmed, Eric P Xing, and William W Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550, 2008.

Feng Nan, Ramesh Nallapati, Zhiguo Wang, Cicero Nogueira dos Santos, Henghui Zhu, Dejiao Zhang, Kathleen McKeown, and Bing Xiang. Entity-level factual consistency of abstractive text summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2727–2733, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.235. URL https://aclanthology.org/2021.eacl-main.235.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1747–1759, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1158. URL https://www.aclweb.org/anthology/N18-1158.

Ani Nenkova and Kathleen McKeown. A Survey of Text Summarization Techniques. In Charu C. Aggarwal and ChengXiang Zhai, editors, *Mining Text Data*, pages 43–76. Springer US, Boston, MA, 2012. ISBN 978-1-4614-3223-4. doi: 10.1007/978-1-4614-3223-4_3. URL https://doi.org/10.1007/978-1-4614-3223-4_3.

Anna Nikiforovskaya, Nikolai Kapralov, Anna Vlasova, Oleg Shpynov, and Aleksei Shpilman. Automatic generation of reviews of scientific papers. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 314–319, December 2020. doi: 10.1109/ICMLA51294.2020.00058.

Bethesda (MD): National Library of Medicine. Pmc open access subset [internet]. https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/, 2003. Accesses: 2022-07-30.

Samir Okasha. Philosophy of science: very short introduction: Oxford university press, 2016.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. Unsupervised learning of sentence embeddings using compositional n-gram features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 528–540, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1049. URL https://www.aclweb.org/anthology/N18-1049.

Bart Penders. Ten simple rules for responsible referencing. *PLOS Computational Biology*, 14(4):e1006036, Apr 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006036. URL http://dx.doi.org/10.1371/journal.pcbi.1006036.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

Jonathan Pilault, Raymond Li, Sandeep Subramanian, and Chris Pal. On extractive and abstractive neural document summarization with transformer language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9308–9319, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/

v1/2020.emnlp-main.748. URL https://www.aclweb.org/anthology/2020.emnlp-main.748.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. New Jersey, USA, 2003.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

Tarek Saier and Michael Färber. unarXive: a large scholarly data set with publications' full-text, annotated in-text citations, and links to metadata. *Scientometrics*, 125(3):3085–3108, December 2020. ISSN 1588-2861. doi: 10.1007/s11192-020-03382-z. URL https://doi.org/10.1007/s11192-020-03382-z.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2019. URL https://arxiv.org/abs/1910.01108.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. doi: 10.1109/CVPR.2015.7298682.

Darsh J Shah and Regina Barzilay. Generating related work, 2021. URL https://arxiv.org/abs/2104.08668.

Neil R Smalheiser, Dean P Fragnito, and Eric E Tirk. Anne o'tate: Value-added pubmed search engine for analysis and text mining. *PloS one*, 16(3):e0248335, 2021.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Trevor Strohman, W Bruce Croft, and David Jensen. Recommending citations for academic papers. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 705–706, 2007.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 3104–3112. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Oguzhan Tas and Farzad Kiyani. A survey automatic text summarization. *PressAcademia Procedia*, 5(1):205–213, 2007.

Mauri Valtonen, Joanna Anosova, Konstantin Kholshevnikov, Aleksandr Mylläri, Victor Orlov, and Kiyotaka Tanikawa. From newton to einstein: The discovery of laws of motion and gravity. In *The Three-body Problem from Pythagoras to Hawking*, pages 31–49. Springer, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus,

S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Ellen M. Voorhees. The trec-8 question answering track report. In *In Proceedings of TREC-8*, pages 77–82, 1999.

Alex Wang, Kyunghyun Cho, and Mike Lewis. Asking and Answering Questions to Evaluate the Factual Consistency of Summaries. *arXiv:2004.04228 [cs]*, April 2020. URL http://arxiv.org/abs/2004.04228. arXiv: 2004.04228.

Pancheng Wang, Shasha Li, Haifang Zhou, Jintao Tang, and Ting Wang. An analytical study on a benchmark corpus constructed for related work generation. *Lecture Notes in Computer Science*, page 421–432, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-32233-5_33. URL http://dx.doi.org/10.1007/978-3-030-32233-5_33.

Yifan Wang, Yiping Song, Shuai Li, Chaoran Cheng, Wei Ju, Ming Zhang, and Sheng Wang. DisenCite: Graph-Based Disentangled Representation Learning for Context-Specific Citation Generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11449–11458, June 2022. doi: 10.1609/aaai.v36i10.21397. URL https://ojs.aaai.org/index.php/AAAI/article/view/21397.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL https://doi.org/10.1007/BF00992696.

R. F. Woolson. *Wilcoxon Signed-Rank Test*, pages 1–3. John Wiley & Sons, Ltd, 2008. ISBN 9780471462422. doi: https://doi.org/10.1002/9780471462422.eoct979. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780471462422.eoct979.

Jia-Yan Wu, Alexander Te-Wei Shieh, Shih-Ju Hsu, and Yun-Nung Chen. To-

wards generating citation sentences for multiple references with intent control, 2021. URL https://arxiv.org/abs/2112.01332.

Yuxiang Wu and Baotian Hu. Learning to extract coherent summary via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Wen Xiao and Giuseppe Carenini. Extractive Summarization of Long Documents by Combining Global and Local Context. *arXiv:1909.08089 [cs]*, September 2019. URL http://arxiv.org/abs/1909.08089. arXiv: 1909.08089.

Xinyu Xing, Xiaosheng Fan, and Xiaojun Wan. Automatic Generation of Citation Texts in Scholarly Papers: A Pilot Study. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6181–6190, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.550. URL https://aclanthology.org/2020.acl-main.550.

Xinnuo Xu, Ondřej Dušek, Jingyi Li, Verena Rieser, and Ioannis Konstas. Fact-based content weighting for evaluating abstractive summarisation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5071–5081, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.455. URL https://aclanthology.org/2020.acl-main.455.

Libin Yang, Zeqing Zhang, Xiaoyan Cai, and Tao Dai. Attention-based personalized encoder-decoder model for local citation recommendation. *Computational Intelligence and Neuroscience*, 2019:1–7, Jun 2019. ISSN 1687-5273. doi: 10.1155/2019/1232581. URL http://dx.doi.org/10.1155/2019/1232581.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In

H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/zhang20ae.html.

Xingxing Zhang, Furu Wei, and Ming Zhou. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5059–5069, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1499. URL https://www.aclweb.org/anthology/P19-1499.

Yizhe Zhang, Siqi Sun, Xiang Gao, Yuwei Fang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. Retgen: A joint framework for retrieval and grounded text generation modeling, 2021. URL https://arxiv.org/abs/2105.06597.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.552. URL https://www.aclweb.org/anthology/2020.acl-main.552.

Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. Neural document summarization by jointly learning to score and select sentences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–663, Melbourne,

Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1061. URL https://aclanthology.org/P18-1061.

Publication List

Peer-reviewed Publications

Gu N, Ash E, Hahnloser R. **MemSum: Extractive Summarization of Long Documents Using Multi-Step Episodic Markov Decision Processes**[C]//*Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).* 2022: 6507-6522.

Gu N, Gao Y, Hahnloser R H R. **Local Citation Recommendation with Hierarchical-Attention Text Encoder and SciBERT-Based Reranking**[C]//*European Conference on Information Retrieval.* Springer, Cham, 2022: 274-288.

Gao Y, **Gu N**, Lam J, et al. **Do Discourse Indicators Reflect the Main Arguments in Scientific Papers?**[C]//*Proceedings of the 9th Workshop on Argument Mining.* 2022: 34-50.

Gauerhof L, **Gu N**. **Reverse variational autoencoder for visual attribute manipulation and anomaly detection**[C]//*2020 IEEEWinter Conference on Applications of Computer Vision (WACV).* IEEE, 2020: 2103-2112.

O. Gökce, J. Prada, N.I. Nikolov, **N. Gu**, R.H.R. Hahnloser. **Embedding-based Scientific Literature Discovery in a Text Editor Application**. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics* (2020).

Working Papers

Gu N, Hahnloser R H R. **Controllable Citation Text Generation**[J]. *arXiv preprint arXiv:2211.07066*, 2022.

Gu N, Hahnloser R H R. **SciLit: A Platform for Joint Scientific Literature Discovery, Summarization and Citation Generation.**

Emmanuel A. Bauer, Dominik Stammbach, **Nianlong Gu** and Elliott Ash. **Legal Extractive Summarization of U.S. Court Opinions.**

Patents

Gauerhof L, **Gu N**. **Method and device for training a machine learning system**: *U.S. Patent Application* 17/610,669[P]. 2022-8-4.

Gauerhof L, **Gu N**. **Method and device for testing the robustness of an artificial neural network**: *U.S. Patent Application* 17/596,126[P]. 2022-7-14.