4-23-2021 10:30 AM

A Lightweight and Explainable Citation Recommendation System

Juncheng Yin, *The University of Western Ontario*

Supervisor: Ling, Charles X., *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

The increased pressure of publications makes it more and more difficult for researchers to find appropriate papers to cite quickly and accurately. Context-aware citation recommendation, which can provide users suggestions mainly based on local citation contexts, has been shown to be helpful to alleviate this problem. However, previous works mainly use RNN models and their variance, which tend to be highly complicated with heavy-weight computation. In this thesis, we propose a lightweight and explainable model that is quick to train and obtains high performance. Our model is based on a pre-trained sentence embedding model and trained with triplet loss. Quantitative results on the benchmark dataset reveal that our model achieves impressive performance with or without metadata. Qualitative evidence shows that our model pays different levels of attention to adequate parts of citation contexts and metadata, suggesting that our method is explainable and more trustable.

Keywords: Citation Recommendation, Deep learning.

Summary for Lay Audience

In recent years, natural language processing has witnessed tremendous breakthroughs in different research problems such as machine translation and word processor. These improvements have dramatically changed human's lifestyles and make our life much easier. However, there are still many areas that need to be explored. The citation recommendation, especially the local citation recommendation, is just one of these areas. Considering the increased number of publications in recent years, it becomes harder and harder for researchers to find appropriate papers to cite nowadays. The local citation recommendation is aimed at solving this problem.

The local citation recommendation just imitates human's way of thinking. Given the citation contexts of several sentences, the local citation recommendation can provide the user possible papers to be cited. Then the user can choose from these papers, which dramatically reduce the user's workload. In this thesis, we focus on the local citation recommendation problem. We propose an innovative method based on a pre-trained sentence encoder. Our method outperforms the baselines in all metrics.

Acknowledgements

To my supervisor, Dr. Charles Ling, I owe an immense debt of gratitude for his support, mentorship, scientific insights and contagious enthusiasm during my studies. His consistent, patient confidence in me was essential in the performance of the work described in this thesis. He provides much help since I came to Canada, I believe he considerably exceeded the expectations of his role as supervisor.

I would like to thank Tanner, Xinxu, Yuanyuan, Yining, and Jinhang for their encouragement and advice during this research and my life in Canada.

I would also like to show special gratitude to my parents for their constant support throughout my research and life.

Table of Contents

1. Introduction

In this chapter, we provide some background knowledge about the citation recommendation problem. We also discuss the importance of this research area and our contribution toward solving this citation recommendation problem. This thesis outline is introduced at the end.

1.1 Motivation

With the rapid development of scientific research, there are much more papers published every year in different research areas. The rich amount of papers makes it more and more time consuming for researchers to find appropriate papers to cite when they write their papers (Küçüktunç, Onur, et al. 2014). Therefore, automatic citation recommendation, which automatically gives researchers suggestions for papers to cite, shows a very good application prospect.

There are two main approaches of citation recommendation (Huang, Wenyi, et al. 2015). One approach is global citation recommendation, which uses information from the whole written paper to give suggestions. This approach is similar to the recommendation system, mainly relying on citation relationships which are already known to provide suggestions (Beel, Joeran, et al. 2016). Recent works also use partial text information of the manuscripts (Bhagavatula, Chandra, et al. 2018). The other approach is local citation recommendation, which mainly uses information of several sentences around the paper citation location (which we call *citation contexts*) and some information of papers potential to be cited (which we call *candidate papers*). Metadata, such as author names, venues, and published years, can also be used to improve performance in both approaches (Ebesu et al. 2017; Bhagavatula, Chandra, et al. 2018).

The global approach uses the whole manuscript information to make citation recommendations. Compared to the local approach, it has more information to analyze. Thus, it can make more general recommendations. Recommendations given by the global approach tend to be more related to the central ideas of citing paper. However,

due to the difficulty of processing the semantic information of the whole manuscripts, most global approach works only use citation relationships already provided in the training dataset (Beel, Joeran, et al. 2016) and partial text (e.g., abstract) to make recommendations. This limits the recommendation accuracy. On the other hand, the local approaches mainly suggest candidate papers based on citation contexts with the scope of several surrounding sentences (Huang, Wenyi, et al. 2015). Therefore, the suggested candidate papers tend to be more related to the specific contexts. Traditionally, before deep learning, a variety of methods have been used to solve this local recommendation problem, such as Restricted Boltzmann Machines (Tang and Zhang 2009), collaborative filtering (Liu, Haifeng, et al. 2015) and statistical machine translation (Tang, Xuewei, et al. 2014). However, the outcome is not completely satisfactory. Recently, neural networks have also been used in this area, significantly improving the performance. Huang et al. firstly use a feedforward neural network to learn embeddings of words in the citation contexts and document embeddings. This method only uses a bag of words as input and ignores the semantic information of the sequence. Ebesu et al. use the Time-Delay Neural Networks (TDNNs) (Collobert, et. al. 2008) and RNN to extract information from citation contexts and candidate papers. Attention mechanism is also used to better utilize information extracted from the TDNNs. They also encode author names of both citation contexts and candidate papers as metadata into the network, which dramatically enhances the performance. However, this trick makes the recommendation much favor of self-citation (the writer citing their own or their colleagues' papers), which makes it less useful when researchers want to obtain suggestions for candidate papers they do not know. Some other papers also use metadata such as venues and published dates (Bhagavatula et al. 2018). However, this also introduces biases. Due to the widely performed peer-review process, papers that are published in the same venue as the citing papers are more often selected as citations (Färber, M. et al. 2020).

1.2 Contribution

As we discussed earlier, it is quite common for researchers to cite their own papers, or papers from their colleagues or the same venues. Although researches show that this is not totally harmful (Tahamtan, I. et al. 2018), an ideal citation recommendation method

should be independent of any bias. In this paper, we propose an innovative neural network model for local citation recommendation, which mainly uses the citation contexts around potential citing location to give suggestions. Since it is based on semantic information analysis of citation contexts, it prevents possible biases caused by authors' preference. Specifically, we use three sentences before the potential citing location and two sentences after the potential citing location, which is similar to human attention span. This method has also been verified by our experiments to be the best. For candidate papers, we extract information from their titles and abstracts. If title and abstract of citation contexts are given, they can also be used in our model to further improve performance. We use a pre-trained sentence embedding model to extract information from both citation contexts and candidate papers. Since the pre-trained model is not designed for this specific task, its embedding outcome is not very accurate. Moreover, the embeddings for citing and cited papers have different dimensions. Therefore, we use re-embedding layers to remap the embeddings for citing and cited papers to the same new vector space. After that we compute the cosine similarity between the embeddings for citing and cited papers. We use the cosine similarity scores to measure the correlation between the citing papers and recommended cited papers. The higher the scores, the better the recommendations.

We train our network on a benchmark citation dataset (Jeong, Chanwoo et al. 2019), which contains 4898 papers published from 2007 to 2017 as well as 17274 citation contexts. Our method reaches a pretty good score and improves performance on benchmark by a clear margin compared to competitive baselines.

Our model has some key features: 1. Lightweight. The proposed model is based on a pre-trained sentence embedding encoder, which dramatically enhances the time and compute efficiency compared with other models. 2. Explainable. Our model automatically learns to assign different weights to adequate parts of the citation contexts, which intuitively shows us the contribution of different parts of citation contexts. 3. Content based. Our model is able to achieve high performance solely based on semantic information. 4. High performance. Experimental results on the benchmark dataset demonstrate our model produces a significant improvement on Recall and Mean Reciprocal Rank (MRR) compared with competitive baseline models.

We are currently building an automatic citation recommendation system based on our model as illustrated in Figure 1.1. When the user inputs the citation contexts, the system will automatically suggest possible places to add citations, denoted by [?]. When the user clicks on [?], a list of recommended papers is provided. The user can choose papers from the list to see more concrete information. After the user selects a paper (or papers) to cite, the model will update citation recommendations for the nearby sentences dynamically. The system is lightweight as it can quickly respond to users' selection on the citation recommendation, and easy to update when new publications are added to the dataset.

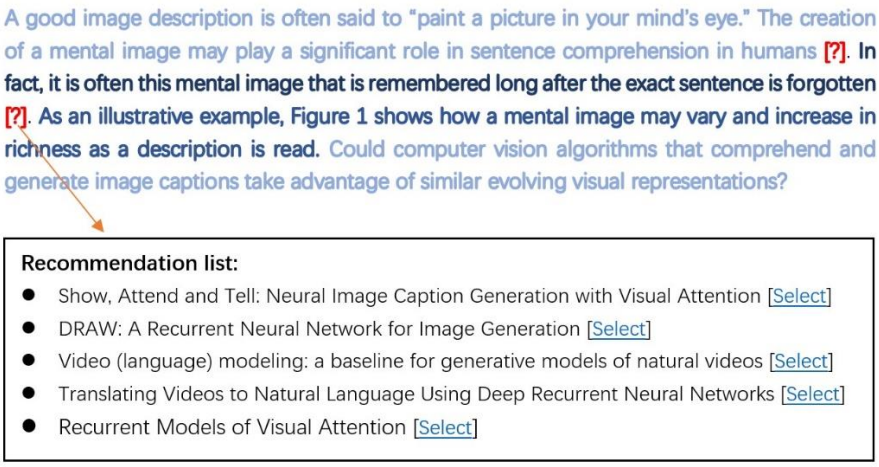

Figure1.1: An illustration of our paper citation recommendation system. Different colors for the citation contexts demonstrate the various levels of attention paid by the system. Darker colors mean sentences that raise more attention. The recommendation list shown here is actually generated by our model.

1.3 Thesis Outline

The structure of this thesis is organized as follows. In Chapter 2, we discuss some related work, including TDNN to RNN, attention mechanism, BERT and its variation. In Chapter 3, we describe our innovative approach to solve the local citation recommendation problem. In Chapter 4, we discuss the experiments, including the dataset used in our research, baseline models, and experiment results. In Chapter 5, we demonstrate the practice use example of our model. In Chapter 6, we conclude this thesis and outline potential future work.

2. Background

In this chapter we define some terms used in the research of citation recommendation and review some previous work. We also make a comparison between the global citation recommendation and the local citation recommendation. Besides that, we provide some essential background knowledge needed to be known in this area.

2.1 Terminology

In this section we introduce some important concepts in the area of citation recommendation.

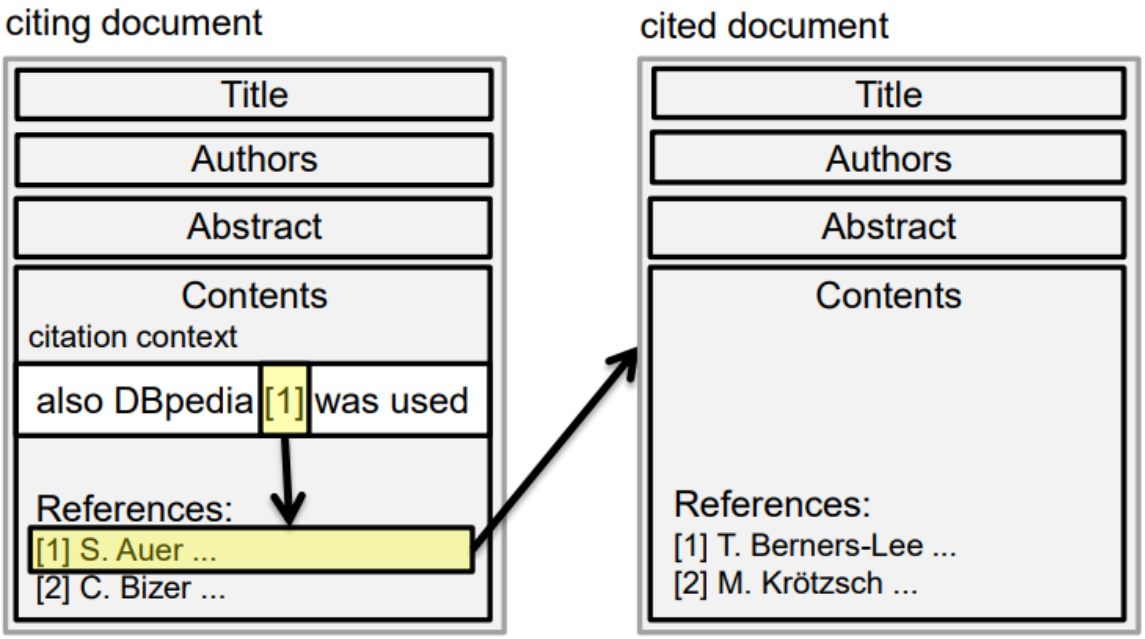

Figure 2.1 Visualization of a citation in a research paper.

As shown in Fig. 2.1, a typical citation is a kind of relationship between two papers. One paper, which we call *citing paper*, cites something from another paper, which we call *cited paper* (Färber, M. et al. 2020). And potential papers to be cited by the citing paper are called *candidate papers*. Both citing paper and candidate paper are composed of title, authors, abstract, contents and some other information. The sentences in the contents of the citing paper, among which contents from the candidate paper are cited, are called

citation contexts. The titles, abstracts, contents are categorized as context information, while the author names, venues, publish years et al. are categorized as metadata. If a research method uses information of the whole citing papers and candidate papers, it is called global citation recommendation, while methods mainly use citation contexts information are called local citation recommendation (or context-based citation recommendation) (Huang, W. et al. 2015).

2.2 Recurrent Neural Network

A recurrent neural network is a class of artificial neural networks where nodes in a directed graph connect to each other along the temporal axis. As we all know, the feed forward neural networks have a strong ability to extract information from the input tensors. However, this kind of neural network is densely connected and the resources and time it costs grow dramatically as the size of the networks grows. This limits its ability to proceed through really long sequences. However, humans often use really long thinking processes. For most of the circumstances, humans do not begin to think about something from scratch, but based on previous knowledge. The recurrent neural network is a derivation from the feed forward neural network. It can use its internal state to save previous information and process really long sequences. This makes the recurrent neural network widely used in different fields, such as speech recognition and language translation.

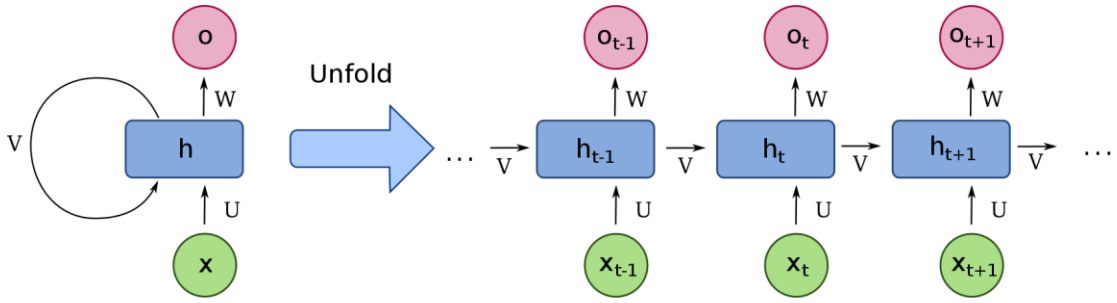

Figure 2.2: Unfolded basic recurrent neural network

Fig. 2.2 shows the classic architecture of the recurrent neural network. The basic recurrent neural network is a recurrence layer containing a series of neuron-like nodes.

Each node connects to itself in the next time step. In each step, each note maintains a different hidden state, and a modifiable real-valued weight. According to the time steps, the nodes can be divided into three categories: the input nodes, which get information from outside of the recurrent neural network; the hidden nodes, which maintain and modify the hidden states obtained from the previous nodes; the output nodes, which yield the result.

2.2.1 Long Short Term Memory networks

As we mentioned before, the ordinary feed forward neural networks cannot remember long term information properly. The basic architecture of recurrent neural networks is proposed to solve this problem. However, this basic model fails to work in supposed performance. Research (Bengio, Y. et al. 1994) has shown that this simple architecture has Inherent defects which makes this basic recurrent neural network model hard to preserve long memory in practice. Therefore, many researchers have been working to find more efficient network architecture. The Long Short Term Memory networks is an outstanding representative of them.

Long Short Term Memory networks is a classic special designed architecture of the recurrent neural network, specifically aiming at solving the long term dependency problem. This kind of networks are usually abbreviated as LSTMs. Since the first research of the LSTM (Hochreiter, S. et al. 1997) is published, many researchers follow this work and get it refined and widely used in different areas.

All the recurrent neural networks have a common basic architecture. They all have some basic modules that are repeatedly used over time. In the basic recurrent neural network models, this kind of modules are usually very simple, such as a single linear layer with tanh activation function.

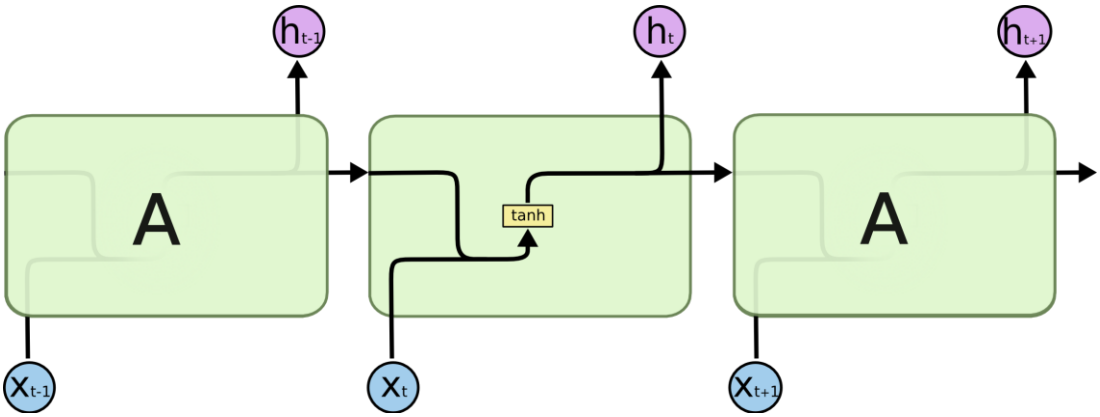

Figure 2.3: The architecture of a basic recurrent neural network model.

The LSTMs also follow this classic architecture. However, instead of using a simple repeated module, the LSTMs design a pretty delicate structure inside the module, with four neural network layers interactive in a clever way.

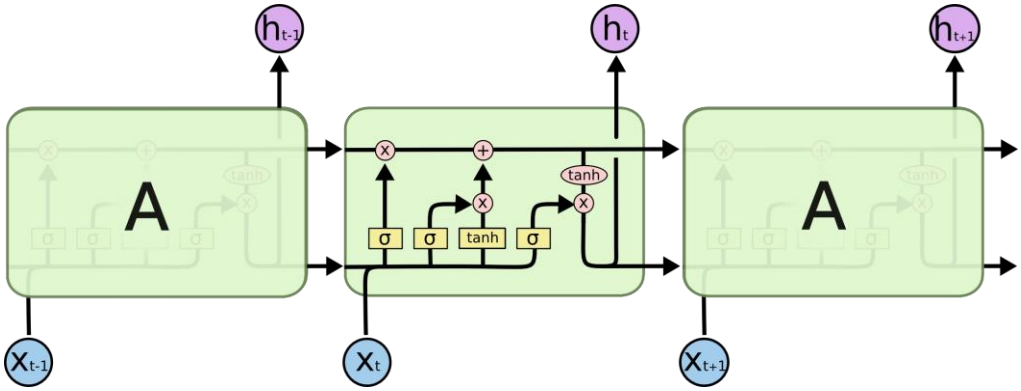

Figure 2.4: The architecture of a LSTM

The most important part of this module is located at the top of this module. The horizontal line with one multiplication and one plus operation. This line denotes the operation over the cell state, which is denoted as C_t. Since it runs through the whole chained network with few changes, it is relatively easy for it to keep information unchanged through the whole process. Moreover, the LSTM also has many special designed structures called gates, which are used to modify the information in the cell states.

The gates are composed of sigmoid layers and multiplications operators. The sigmoid layer computes a probability based on the input information. The probability is between 0

and 1. If the probability of a gate is 0, that gate will block all the information from getting through the gate. On the other hand, if the probability of a gate is 1, that gate will let all input information pass without any modification.

For any time step t, this basic module of the LSTM takes one input feature called x_t from outside of the network. It also inherits the cell state C_{t-1} and hidden state h_{t-1} from the previous repeated module. After a series of operation, it passes the new cell state C_t and hidden state h_t to the next module, and output h_t as output to the outside if necessary.

The first gate is called the forget gate. As its name shown, this gate decides how much to forget from the previous cell state C_{t-1}. This gate takes the previous hidden state h_{t-1} and input feature x_t as inputs, and use a sigmoid layer to map these inputs to a vector of values between 0 and 1. Each value indicates a probability to remember the information from the corresponding digits of the previous cell state C_{t-1}. If the value is 1, we keep the value in that digit of C_{t-1} unchanged. On the other hand, if the value is 0, the information contains in that digit of C_{t-1}will be totally forgotten. This process can be depicted as:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Where W_f and b_f are the trainable parameters for the gate.

The second gate is called the input gate. This gate controls how much input information to be added to the cell state. On the one hand, we project the previous hidden state h_{t-1} and input feature x_t into a tanh layer to output a vector of candidate values $\widehat{C_t}$, which is added to the cell state later. On the other hand, we use a sigmoid layer to compute a probability based on h_{t-1} and x_t. After that, this probability is used to decide how much of the candidate values $\widehat{C_t}$ will actually be added to the cell state. This process can be depicted as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\widehat{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

After these processes, the cell state can be updated. We use f_t to decide how much to forget from the previous cell state C_{t-1}, and use i_t to decide how much to add from the new computed candidate cell state C_t. Finally, we add them together to get the new cell state C_t:

$$C_t = f_t * C_{t-1} + i_t * C_t$$

Beside computing the new cell state and keeping and passing it along time steps, we may also want to output some information to the outside. We achieve this by using a modified version of the cell state. First we pass the original cell state C_t to a tanh layer to project the value to -1 and 1. Then we use a sigmoid layer to compute a probability vector based on h_{t-1} and x_t, which is denoted as o_t. Finally, we use o_t to control how much we output form the cell state. This is depicted as:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * tanh(C_t)$$

The hidden state h_t is passed into next module along time as same as the cell state C_t. At the same time, h_t can also be output to the outside as the final output.

2.2.2 Gated Recurrent Unit

The gated recurrent unit, i.e. GRU, is a simplified version of LSTM, which was introduced by Kyunghyun Cho et al. in 2014. Compared to the LSTM, the GRU has fewer parameters. For example, it does not have an output gate. Due to its simplified structure, it is lighter weighted thus more quickly to learn. On the other hand, some research shows that its ability to handle long sequences is weaker compared to the LSTM. (Weiss, G. et al. 2018; Britz, D. et al. 2017)

The Fig. 2.5 depicts a classic architecture of the GRU unit. For every step t, we have the input vector x_t, previous hidden state h_{t-1}, output hidden state h_t. The output hidden state h_t can also be used as the final output, i.e. y_t.

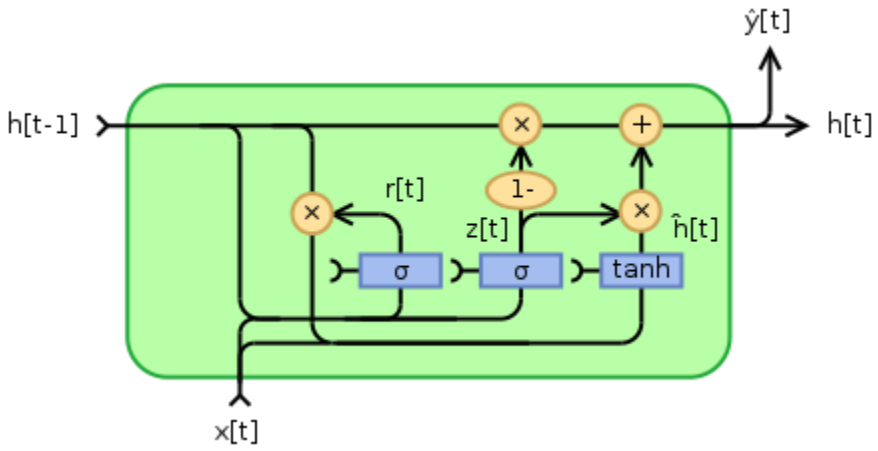

Figure 2.5: architecture of the gated recurrent unit.

The process of the GRU can be depicted as follows:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$

$$\widehat{h_t} = \phi_h(W_h x_t + U_h(r_t \cdot h_{t-1}) + b_h)$$

$$h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \widehat{h_t}$$

Where σ_g is the sigmoid function, ϕ_h is the tanh function, and $W_z, W_r, W_h, U_z, U_r, U_h, b_z, b_r, b_h, z_t$ is the trainable parameters.

2.3 Word Embedding

Word embedding is one of the fundamentals for the natural language process. To process the sequences of words using neural networks, we need to map the words to matrices of real numbers. The word embedding just provides us this approach.

2.3.1 Word2vec

Word2vec (Mikolov, T. et al. 2013) is one of the most popular methods for word embedding. This method uses shallow neural networks to learn word embeddings automatically. Therefore, it can be quickly achieved and is rather efficient. Moreover, this method is a statistical method, and it is a standalone word embedding not aiming at a

specific task. Thus the word embeddings learned by this method can be used directly by loads of other natural language tasks.

The Word2vec first maps the words to one-hot vectors with the size of all vocabularies. Since the one-hot vectors are independent of each other, we cannot find any relationship of different words based on the one-hot vectors. Therefore, Word2vec projects the one-hot vectors to distributed representations afterwards. Intuitively, the distributed representations are supposed to reflect the meaning or relationships of words. Words that have similar meanings should be close to each other. This is measured by the cosine similarity for the distributed representations in the new vector spaces.

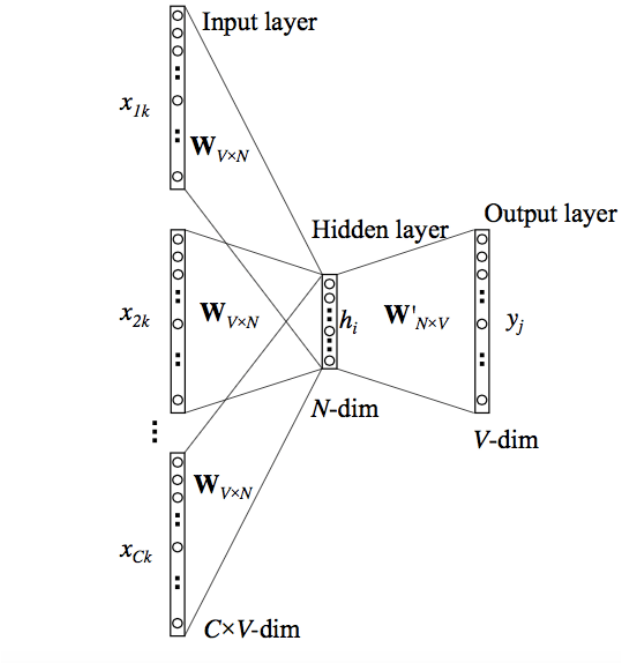

Figure 2.6: The architecture of the CBOW model.

There are two models for doing this. Common Bag of Words (CBOW) and Skip Gram. The CBOW uses words in a context to predict a specific word. As depicted in Fig. 2.6, each input word is denoted as a V-dimension vector, where V is the vocabulary size. The N-dimension hidden layer only does the matrix multiplication without any activation function. This operation maps the input embeddings to a new vector space. Then the V-

dimension output layer remaps the outputs from the hidden layer to V-dimension, and uses a softmax activation function to map its values to 0~1. This denotes the probability distribution for the target word along the vocabulary table. The model takes C words as input in total. These C words are projected by the hidden layer to the new vector spaces. We take their average as the input to the softmax layer. After training the neural network, the hidden layer is used to map V-dimension one-hot word vectors to N-dimension distribution representations, which are the word embeddings we need.

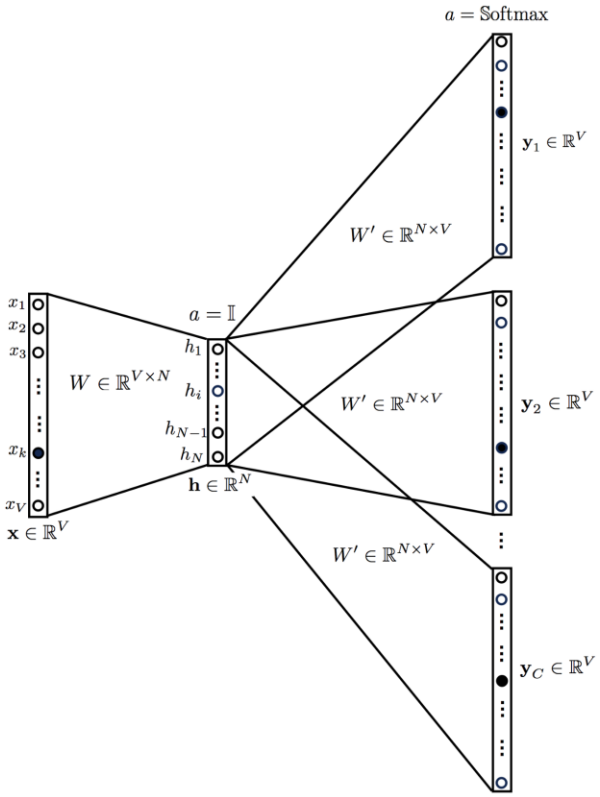

Figure 2.7: Architecture of the Skim-Gram model

The Skim-Gram model is the other one to obtain the projection. It is just like the reverse version of the CBOW model to some content. It uses one word as input and predicts the contexts around this word. As depicted in Fig. 2.7, the input is the one-hot word representation x with dimension V, where V is the vocabulary size. The hidden layer also only does the matrix multiplication, the same as the CBOW method. It projects the input V-dimension x to the N-dimension vector space. The output layers are C separate softmax layers. Each layer predicts one word. After training the neural network, we also

use the hidden layer to map words to N-dimension distribution representations, which is the word embeddings we need.

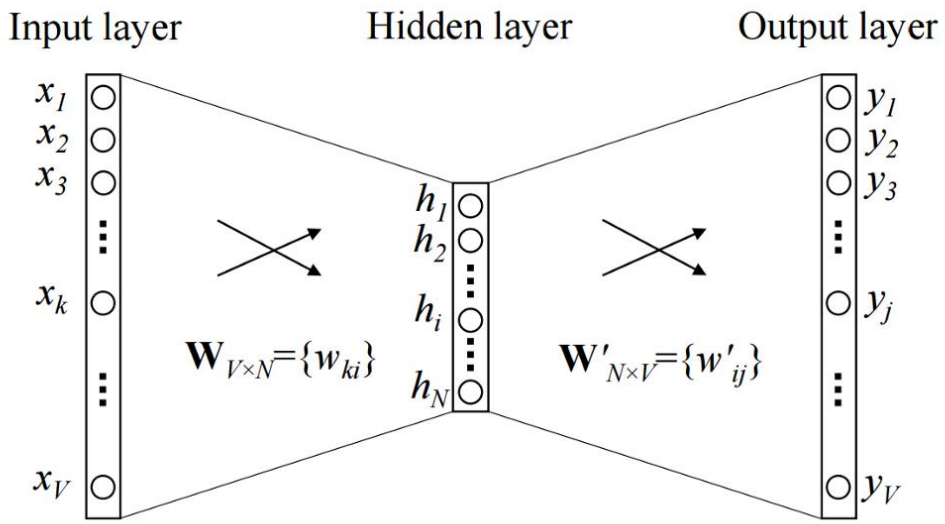

Figure 2.8: Architecture of the simplified model.

The two models we discussed earlier are both widely used. Researchers keep modifying and improving these two models. Now we get an extremely simplified method to compute the word embedding. For any given contexts, we just randomly pick up one word as the input, and use this to predict a word nearby. Thus the input is a one-hot word vector with the dimension of V, where V is the vocabulary size. The hidden layer maps the input to N-dimensional vector space via matrix multiplication. Finally, the output layer is a softmax layer with dimension of V, which maps the output from the hidden layer to the probability distribution for the target word along the vocabulary table.

After training, we also use the trained hidden layer to calculate the distribution representations for input words as the word embeddings. This method is extremely simple, but it turns out to be pretty efficient.

2.3.2 GloVe

The Global Vectors for Word Representation (Pennington, J. et al. 2014), or GloVe, algorithm is based on the word2vec method and makes some improvement on it to make

the learning process more efficient. The word2Vec method only uses the local statistical information to learn the word embeddings, while the GloVe method also utilizes the global information.

The GloVe is based on the idea of the co-occurrence matrix. This method believes that the co-occurrence matrix is a good source of the semantic information. Assuming we have a corpus with V different words, we define the set of all words to be $Word = \{word_1, word_2 \ldots word_V\}$. The co-occurrence matrix is defined as $X \in \Re^{V \times V}$. let the X_{ij}, i.e. the i^{th} row and j^{th} column of the matrix X, denotes times for $word_i$ and $word_j$to show up together.

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

Figure 2.9: The co-occurrence matrix for the sentence “the cat sat on the mat”.

Fig. 2.9 depicts an example of the co-occurrence matrix. Assume the corpus only contains one sentence: the cat sat on the mat. The word “the” shows up with “cat”, “on” and “mat”. Thus X_{12}, X_{14}, X_{15} are all equal to 1 in the first line of the co-occurrence matrix. The same goes for the other rows. It is worth noting that the co-occurrence matrix is a symmetric matrix.

For three randomly picked up words $word_i, word_j, word_k$ We define P_{ik} as the probability to see word i and k together:

$$P_{ik} = \frac{X_{ik}}{X_i}$$

Where X_{ik} denotes the number of times for $word_i$ and $word_k$ to show up together. X_i denotes the total number of times for $word_i$ to show up in the corpus.

Now we can use P_{ik}/P_{jk} to denote the semantic similarity between $word_i$ and $word_j$ based on the third word $word_k$(i.e. the probe word). If $word_k$ is highly connective to $word_i$ but irrelevant to $word_j$, P_{ik}/P_{jk}will be very large. If $word_k$ is irrelevant to $word_i$ but highly connective to $word_j$, P_{ik}/P_{jk}will be close to 0. If the similarity between $word_i$ and $word_k$ and between $word_j$ and $word_i$is similar, P_{ik}/P_{jk}will be close to 1. The Fig. 2.10 shows a concrete example.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Figure 2.10: The behavior of P_{ik}/P_{jk} for various words

The GloVe method uses neural networks to learn the embeddings. We define two different embedding layers w and u. w is for the $word_i$ and $word_j$, while u is for the $word_k$. We use $F(i,k) = e^{{w_i}^T u_k}$ to emulate P_{ik}. Therefore we have $F(i,k) = e^{{w_i}^T u_k} = P_{ik}$, $F(j,k) = e^{{w_j}^T u_k} = P_{jk}$, $F(i-j,k) = e^{{w_{i-j}}^T u_k} = e^{{w_i}^T u_k}/e^{{w_j}^T u_k} = F(i,k)/F(j,k)$. Moreover, we have ${w_i}^T u_k = log(P_{ik}) = log(X_{ik}) - log(X_i)$, ${w_i}^T u_k + log(X_i) = log(X_{ik})$. We define b_w, b_u to be the biases of the embedding layers w and u, and use them to emulate the $log(X_i)$. Thus we have ${w_i}^T u_k + b_w + b_u = log(X_{ik})$. Or ${w_i}^T u_k + b_w + b_u - log(X_{ik}) = 0$.

Therefore, the GloVe method defines the loss function as:

$$J = f(X_{ij})({w_i}^T u_k + b_w + b_u - log(X_{ik}))^2$$

Where $f(X_{ij}) = (x/x_{max})^a \; if \; x < x_{max} \; else \; 0$.

2.3.3 Deep contextualized word representations

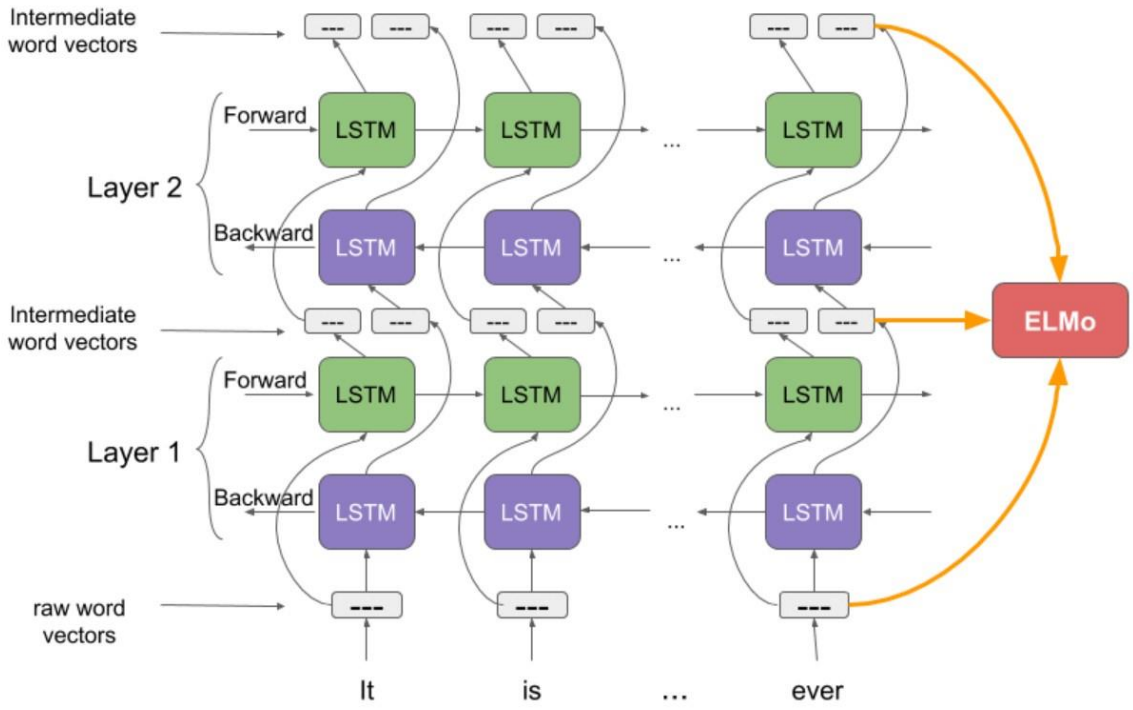

Figure 2.10: The architecture of the ELMo.

In 2018, Peters, M. E. et al. proposed a new deep contextualized word representation method, which is also called ELMo. Unlike previous word embedding methods that calculate fixed embedding values for each word, such as the word2vec, GloVe and fastText(Bojanowski, P. et al. 2017). The ELMo method generates the word embeddings based on the whole sentence. They first train multilayer bidirectional LSTMs with a coupled language model objective on a really large training dataset. After that, they extract the inner states vector of the LSTMs and concatenate them together. Then they use a linear layer neural network to learn the combination of the inner states. They achieve really impressive performance by this approach.

As depicted in their work, different levels of LSTM inner state capture different levels of information from the texts. The lower LSTM tends to focus on information about syntax while the upper LSTM tends to learn high level information such as the word meanings in

specific contexts. The ELMo simultaneously exposes all these features to further networks of different end tasks and lets them learn their key points automatically. The Fig. 2.10 depicts the overall architecture of the ELMo model.

Suppose we have a series of N words, denoted as $\{t_1, t_2, \ldots, t_N\}$. A forward language model calculates the overall possibility for the sequence to occur as the mutual possibility for each of the N words to occur given the previous words:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1})$$

A backward language model is much similar to the forward model. It also calculates the overall possibility for the sequence based on the mutual possibility. However, the possibility is for each of the N words to occur given the afterwards words:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N)$$

The ELMo method jointly maximize the log likelihood for the forward and backward LSTM models:

$$\sum_{k=1}^{N} (logp(t_k | t_1, \ldots, t_{k-1}; \Theta_x, \Theta_{forwardLSTM}, \Theta_s) + logp(t_k | t_{k+1}, \ldots, t_N; \Theta_x, \Theta_{backwardLSTM}, \Theta_s))$$

Where $\Theta_x$denotes the trainable parameters for the token representation layer, $\Theta_s$denotes the trainable parameters for the softmax layer, $\Theta_{forwardLSTM}$ denotes the trainable parameters for the forward LSTM model, $\Theta_{backwardLSTM}$ denotes the trainable parameters for the backward LSTM model.

After these works, the ELMo method uses the linear combination of the intermediate state values and changes the weight of different intermediate states based on specific tasks. For each word t_k, a L-layer bidirectional LSTMs model computes a set of $2L + 1$ representations:

$$R_k = \{{x_k}^{LM}, {h_{forward,k,j}}^{LM}, {h_{backward,k,j}}^{LM} | j = 1, \ldots, L\} = \{{h_{k,j}}^{LM} | j = 0, \ldots, L\}$$

Where ${h_{k,0}}^{LM}$ denotes the intermediate state for the token representation layer and ${h_{k,j}}^{LM}$ denotes the concatenation of ${h_{forward,k,j}}^{LM}$ and ${h_{backward,k,j}}^{LM}$.

After all of these, the ELMO concatenates all the $2L+1$ representations in R_k into a single vector. Then it learns the linear combination of R_k. Thus we have:

$$ELMo_k{}^{task} = E(R_k; \Theta^{task}) = \gamma^{task} \sum_{j=0}^{L} s_j{}^{task} {h_{k,j}}^{LM}$$

Where the s^{task} are softmax-normalized weight for the intermediate states, and γ^{task} is a scalar variable used to scale the entire model.

2.4 Attention Mechanism

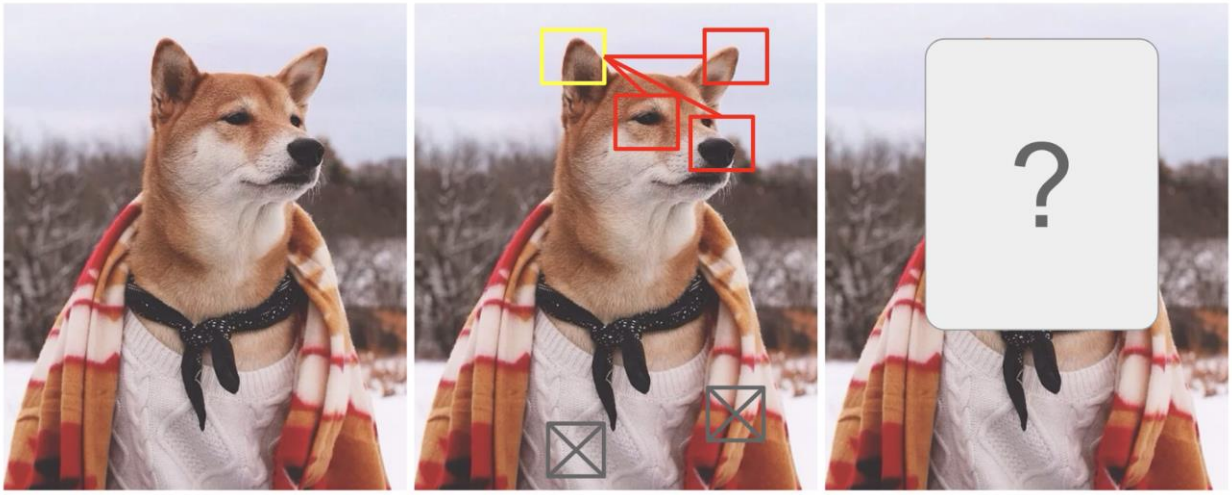

Figure 2.11: A Shiba Inu in a men's outfit

Attention mechanism is a recently proposed method and widely used in machine learning lately. This mechanism is in part inspired by the way humans observe. The Fig. 2.11 demonstrates a concrete example. Looking at a scene, we humans may focus on specific parts with high resolution, such as the dog face, while leaving the other parts in low resolution, such as the forest in the background. We may change the part focused on over time and make some references based on the information we learned from some parts of the image. Given the red boxes in the second picture, we can see the dog's nose,

eye and a right ear. It is reasonable for us to guess that there is also a left ear in the yellow box. However, as depicted in the third picture, it is hard for us to guess what is hidden behind the blank box giving the sweater and blanket.

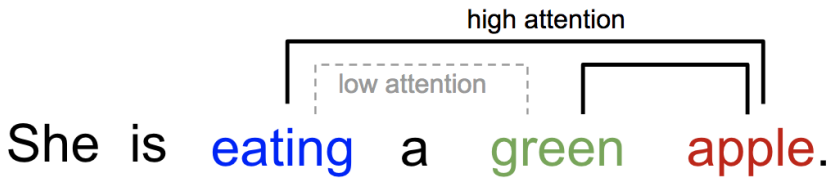

Figure 2.12: The attention distribution along a sentence.

The same circumstances can occur in the sentence reading cases. As depicted in Fig. 2.12. When we see the word “eating”, it is reasonable for us to expect a word indicating some food to follow. Thus we pay great attention to the word “apple” and “eating” together. While the word “green” makes necessary supplements for the description of the word “apple”, it is less relevant to the word “eating” directly.

In a summary, the attention mechanism applies different weights to the raw input features, no matter whether the input is an image or a sentence or something else. This helps downstream networks to focus on more important parts of the raw input.

As we discussed earlier, the RNN networks have its inherent flaws. When used as a sentence encoder, the RNN only uses the last hidden states as the final output to represent the whole sentence. This makes the RNN hard to memorize long distance information. The attention mechanism makes it possible for the final output to peek information directly from the whole raw input sentence. The weights for the attention changed according to different outputs.

The Fig. 2.13 depicts an application of the attention mechanism in the Seq2Seq model (Sutskever, I. et al. 2014). In every time step, the context vector can see all information from the encoder, and learn to focus on different parts automatically. This dramatically boosts the performance of the whole model.

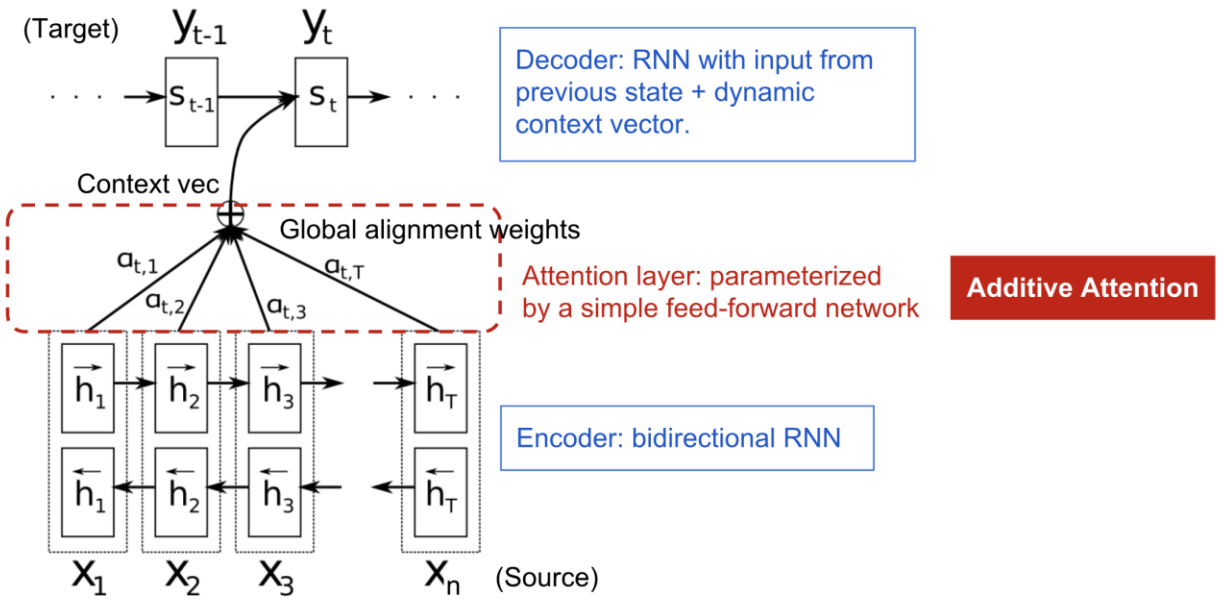

Figure 2.13. The encoder-decoder model with additive attention mechanism (Bahdanau et al., 2014)

Let us consider the application of the attention mechanism in the circumstance of neural machine translation (Bahdanau, D. et al.2014). Suppose we have an input sentences $x \in \Re^n$ and the purpose of our model is to output a sentence $y \in \Re^m$:

$$x = [x_1, x_2, \ldots, x_n]$$

$$y = [y_1, y_2, \ldots, y_m]$$

We use the bidirectional RNN as the encoder. This helps the hidden states $h = [h_1, h_2, \ldots, h_n]$ to remember the information from both the preceding and following words. The decoder hidden states is denoted as $s = [s_1, s_2, \ldots, s_m]$. For each time step t, $s_t = f(s_{t-1}, y_{t-1}, c_t)$, where c_t is called the context vector. c_t is the weighted sum of the encoder hidden states h:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i$$

$$\alpha_{t,i} = align(y_i, x_i) = \frac{exp(score(s_{t-1}, h_i))}{\sum_{i'=1}^{n} exp(score(s_{t-1}, h_{i'}))}$$

The $\alpha_{t,i}$, which is called alignment score in Bahdanau's paper, measures the correlation between the input at time step i and output at time step t. The scores are calculated by a single hidden layer feed-forward neural network. Therefore, the score function can be depicted as follows:

$$score(s_t, h_i) = {v_a}^T tanh(W_a[s_t; h_i])$$

Where v_a and W_a are both trainable parameters of the feed-forward neural network. This network is trained together with the encoder-decoder model. According to Bahdanau's paper, the matrix of alignment scores can well demonstrate the relationship between words from the source and output, as depicted in Fig. 2.14.

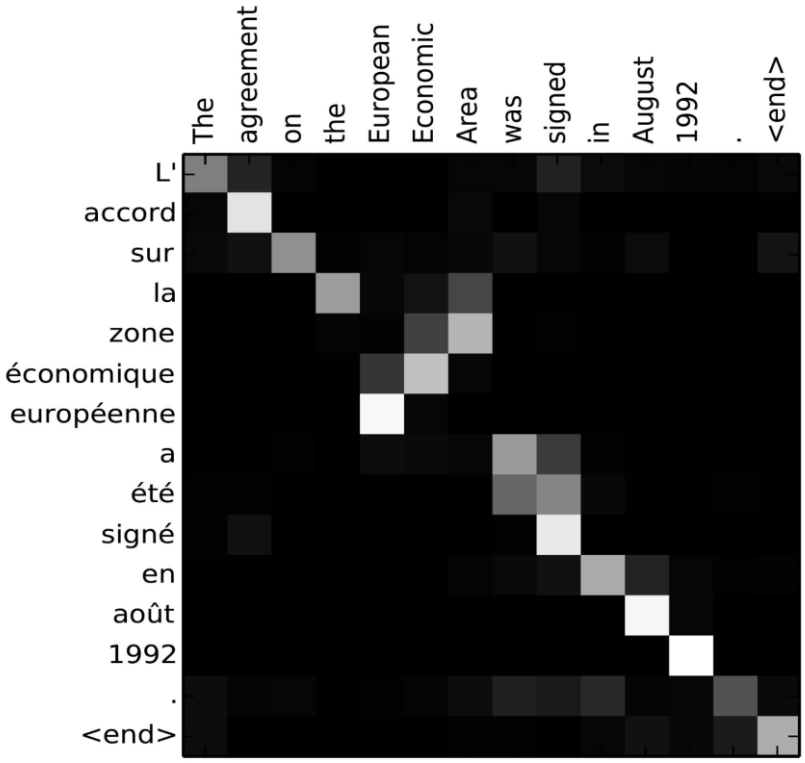

Figure 2.14: Alignment matrix of "L'accord sur l'Espace économique européen a été signé en août 1992" (French) and its English translation "The agreement on the European Economic Area was signed in August 1992". (Bahdanau, D. et al.2014)

The attention mechanism helps researchers make great progress in machine learning visualization. As depicted in Fig 2.15, this machine reading work (Cheng, J. et al. 2016)

uses attention to learn the correlation between current words and previous parts of sentences.

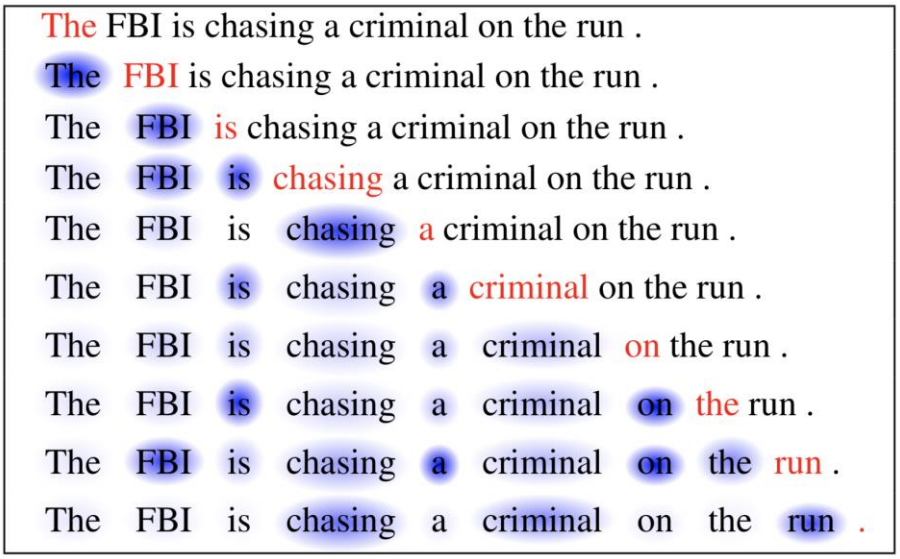

Figure 2.15. The current word is in red and the size of the blue shade indicates the activation level. (Cheng, J. et al. 2016)

2.5 Transformer

The development of the attention mechanism has dramatically boosted the research about sequence models. Vaswani, et al. (2017) proposed a novel architecture to realize Seq2Seq modeling fully based on attention mechanism without the use of recurrent network units.

One of the core ideas of the transformer is the multi-head scaled dot-product attention mechanism. In this module, the raw input sequence is firstly mapped to three different embeddings using three different weight matrices. Thus, we get three matrices V, K, Q. After this we calculates the weight matrix using $V$and K. Finally, we use these new weight matrices to calculate the weighted sum of Q. This process can be depicted as follows:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{n}})V$$

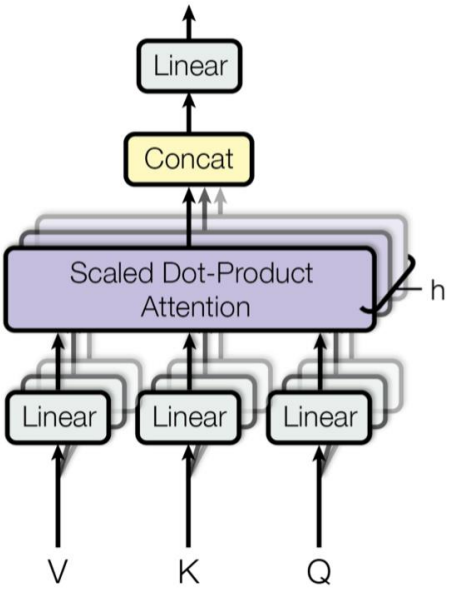

Figure 2.16: Multi-head scaled dot-product attention mechanism

In order to get a more comprehensive understanding of the raw inputs, this process is repeated multiple times. Then all the outputs are concatenated together and projected to a new vector space:

$$MultiHead(Q,K,V) = [head_1;\ldots;head_h]W^O$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Where $W_i^Q, W_i^K, W_i^V and\ W^O$ are learnable parameters.

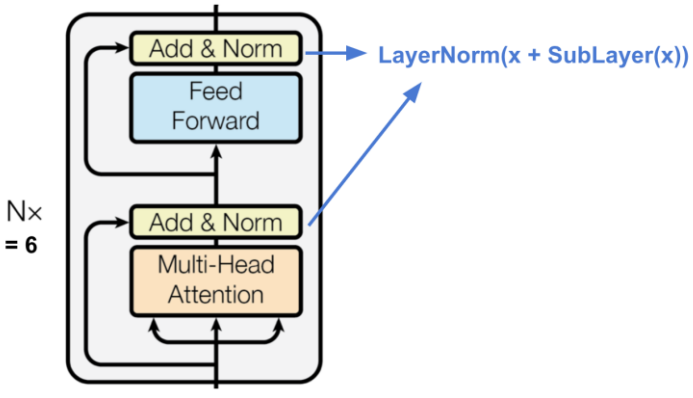

Figure 2.17: The transformer's encoder

The Fig. 2.17 depicts the architecture of the encoder for the transformer. The input feature is first processed by the multi-head attention mechanism then projected to another vector space by the feed-forward neural network. The residual connection and layer normalization are also deployed for each sub-network. This module is repeated for 6 times to get a more comprehensive function.

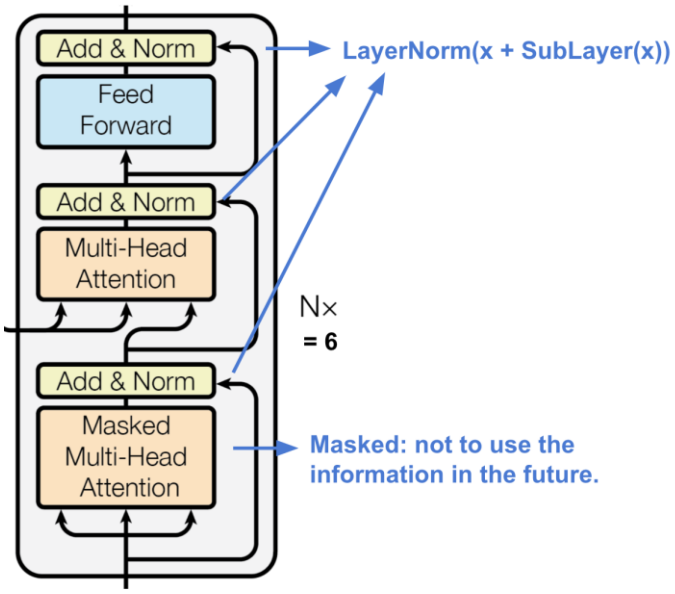

Figure 2.18: The decoder of the transformer.

The Fig. 2.18 depicts the architecture of the decoder. This module contains two multi-head attention blocks and one feed-forward block. The residual connection and layer normalization are also deployed for each block. The first multi-head attention block is modified to get subsequent points in the input feature masked, since we don't want the encoder to peek at the future information of the target sequences when making predictions. The encoder module is also repeated 6 times for a more comprehensive function.

Fig. 2.19 depicts the whole model architecture of the transformer. Both the input and target features are first projected to the same vector space by the embedding layer. a sinusoid-wave-based positional encoding is applied and summed with the embedding output to compact the position information into the features. After the encoding and decoding procedure, the softmax layer is deployed to make the final prediction.

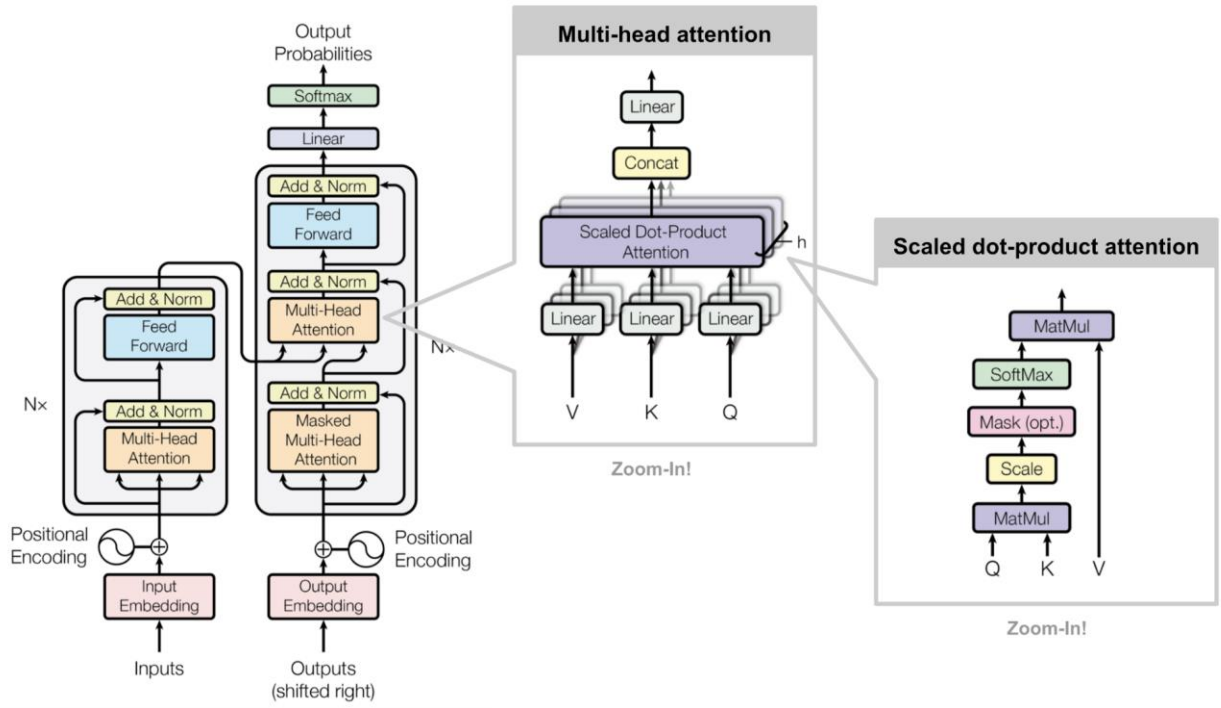

Figure 2.19: the whole model architecture of the transformer.

2.6 Comparison on Recommendation Methods

Methods to provide citation recommendations have been published over the years, using a variety of approaches, and focus on different aspects of this area. Overall, about 50 papers propose some innovative, either global or local citation recommendation methods, among which about a third are local citation recommendation methods (Färber, M. et al. 2020). The global citation recommendation has been exploited by researchers earlier than the local one, probably because that it can be achieved by analyzing the citation relationship graphs, which is unavailable in most cases of the local citation recommendation. With the continuing development of machine learning, researchers can utilize context information better now, which boosts research about local citation recommendations enormously.

In the sections below, we briefly introduced several methods of global citation recommendations. Then we dive into local citation recommendations which are more related to our work.

2.6.1 Global Citation Recommendation

In this section, we briefly walk through several global citation recommendation methods.

2.6.1.1 Collaborative Filtering for Citation Recommendation

Collaborative filtering is a classic method for citation recommendation. This method treats citation recommendations similar to circumstances of e-commerce. Citing papers (customers) make citations of cited papers (items). Similar citing papers (customers) tend to cite similar cited papers (items). Therefore, citing papers making citations of common cited papers can be seen as similar. Their similarity is measured by the common papers that they cite. As depicted in Fig. 2.20, both citing paper i1 and i2 cite the cited paper j2. Thus they are considered to be similar. Citing paper i1 and i4 do not cite some same paper. However, citing paper i1, i2 and i3 cite the same paper j1, while citing paper i2, i3 and i4 cite the same paper j2. Therefore, i1 and i4 can also be considered to have a certain degree of similarity. We can use this citation relationship information to do association mining. Every citing paper can be represented by some other similar citing papers. Therefore, when we make citation recommendations for a specific target paper, we can check which papers are cited by citing papers similar to the target paper.

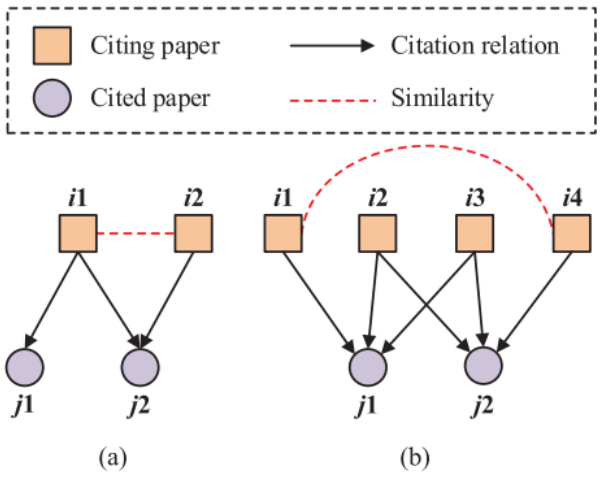

Figure 2.20 Citing papers are similar based on (a) common cited papers or (b) co-occurred citing papers.

2.6.1.2 Content-Based Citation Recommendation

Recent years, some research also uses partial context information and neural networks to do global citation recommendations. Here we briefly introduce a representative study by Bhagavatula, C. et al. Fig. 2.3 depicted the overall architecture of their model.

Their model uses a neural network model to map both the citing papers and cited papers to a new vector space. It is worth noting that they map the citing papers and cited papers to the same vector space, which is much different to researches in the local citation recommendation research area.

As depicted in Fig. 2.3, their citation recommendation method consists of two stages. In the first stage, for a query document d_q(i.e., the citing document), their model embeds it to a document embedding vector space. They use the K nearest neighbors (K = 4 here) as the candidate selection, which is the collection of possible cited papers. In Fig. 2.21 these are d_2, d_3, d_6, d_4. Since $d_7$is very close to the selected range, and that $d_7$is cited by d_3, $d_7$is also added to the candidate selection. In the second stage, they use another neural network model to compute a recommendation score for every candidate paper. The rerank all the candidate selections according to the scores from high to low. After that, they can give the top k recommendations.

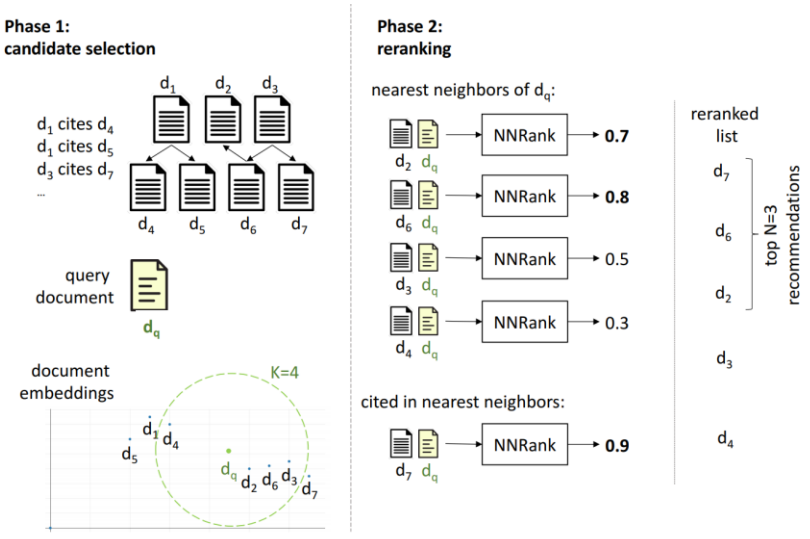

Figure 2.21 An overview of the architecture of the system architecture proposed by Bhagavatula, C. et al.

Their citation recommendation method uses the titles and abstracts of documents to compute the embeddings. For each textual field, i.e. title or abstract, they use the bag of word representation and compute the weighted average of word embeddings as the feature vectors. For any document *d*:

$$f_{d[title]} = \sum_{t \in d[title]} w_t^{mag} \frac{w_t^{dir}}{\left|\left|w_t^{dir}\right|\right|_2}$$

$$f_{d[abstract]} = \sum_{t \in d[abstract]} {w_t}^{mag} \frac{{w_t}^{dir}}{||{w_t}^{dir}||_2}$$

Where ${w_t}^{dir}$is the word direction embedding and ${w_t}^{mag}$ is the magnitude for a word *t.* After that, they normalize the embeddings of these two feature vectors and compute a weighted average of them called e_d.

$$e_d = \lambda^{title} \frac{f_{d[title]}}{||f_{d[title]||_2}} + \lambda^{abstract} \frac{f_{d[abstract]}}{||fd[abstract]||_2}$$

They use e_d to as the document embedding for the document d. The ${w_t}^{dir}, {w_t}^{mag}$ and λ are all trainable parameters in their neural networks.

They use the supervised learning to thain their neural networks. Specifically, they use training sets of triplets $< d_q, d^+, d^- >$, where d_q is the citing paper, d^+ is the paper that is actually cited in d_q, $d^-$is the paper that is not cited in d_q. They use the cosine similarity to measure the relativity between the paper citation pairs. This means that d_q and d^+shall have high similarity, while d_q and d^-shall have low similarity. For the loss function, they use the per-instance triplet loss (Wang et al., 2014):

$$loss = max(\alpha + s(d_q, d^-) - s(d_q, d^+), 0)$$

Where $s(d_i, d_j)$is defined as the cosine similarity between papers d_i and d_j, while the margin parameter $\alpha$is a hyperparameter.

It is easy to get the positive training pairs since they are provided by the dataset. However, the negative training pairs need to be chosen much carefully in order to obtain an ideal performance for the trained model. They basically choose the negative pairs via three approaches. Firstly, they randomly choose some papers not cited by the citing papers in the training dataset. Secondly, they choose some papers that are close to the citing papers in the vector space, but not cited by the citing papers. They use cosine similarity to measure the distance in the vector space. Since the vector space is obtained by their model, they can only get negative examples by this approach started from training the second epoch. After training each epoch, they remap the papers in their training dataset to a new vector space based on their model, and obtain the new nearest neighbors for the citing papers. Thirdly, they choose some papers that are cited by papers which are cited by the citing papers, and that are not directly cited by the citing papers.

In the second phase, they use another model that takes the citing paper and cited paper pairs as inputs and compute a score indicating the possibility for the cited papers to be cited by the citing papers.

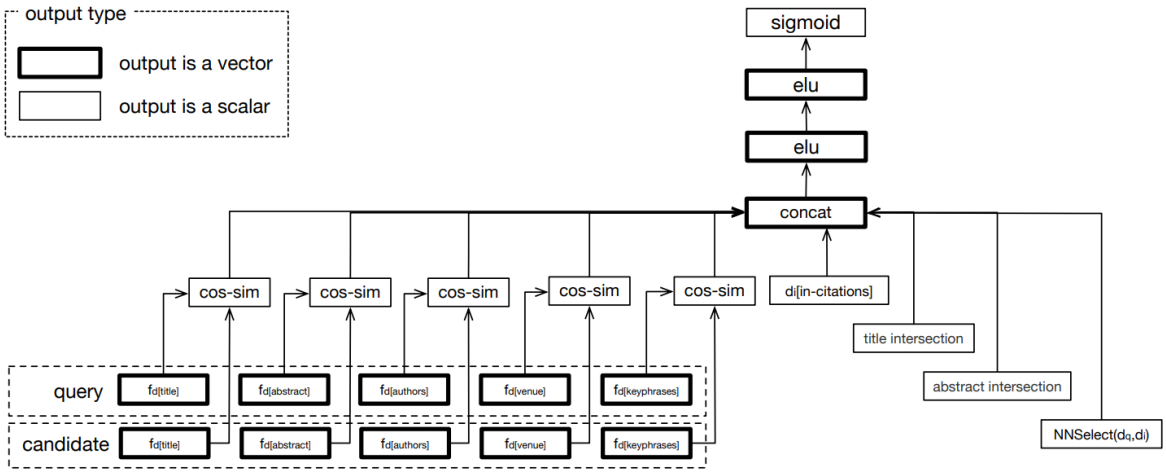

Figure 2.22 Architecture for the computing score model

Fig 2.22 demonstrates the architecture of their computing score model. For each field (if available) in the query and candidate documents, such as the titles, abstracts, authors, venues, and key phrases, they compute the weighted average embeddings of them using

the method we discussed earlier. They also sum the scalar weights of the words that appear in both titles and abstracts of the citing papers and cited papers, and pass this as another input feature. Besides this, they also use the times the cied papers occur in their training dataset as another input feature. They use the logarithm to modify the times to mitigate the impacts of over highly cited papers. After that, they concatenate all these features together as inputs to their final feed-forward neural network to compute the final possibility. Different from the previous phase, here they use the sigmoid function to compute the possibility. This process can be depicted as:

$$s(d_i, d_j) = FeedForward(h)$$

$$h = [g_{title}; g_{abstract}; g_{authors}; g_{venue}; g_{keyphrases}; cos - sim(e_{d_q}, e_{d_i}); \sum_{t \in \cap_{title}} w_t{}^{\cap}; \sum_{t \in \cap_{abstract}} w_t{}^{\cap}; d_{i[in-citations]}]$$

$$g_{field} = cos - sim(f_{d_q[field]}, f_{d_i[field]})$$

2.6.2 Local Citation Recommendation

In this section, we briefly walk through several local citation recommendation methods.

2.6.2.1 the Neural Probabilistic Model

One representative model in the local citation recommendation area is the the Neural Probabilistic Model, i.e. NPM (Huang, W. et al. 2015). This model combines the statistical principles with neural networks.

Suppose the citation context to be c, the cited paper to be d, the goal of local citation recommendation can be seen as find paper d that maximize the probability $p(d|c)$. I.e. the probability to cite paper d given citation context c. According to the Bayes' rule, we have:

$$p(d|c) = \frac{p(c|d)p(d)}{p(c)}$$

Suppose that the citation context c consists of n words $w_1, w_2, w_3 \ldots w_n$, we have:

$$p(c) = p(w_1, w_2, w_3 \dots w_n)$$

The NPM suppose words in the citation context c to be mutually conditional independent, thuw we have:

$$p(w_1, w_2, w_3 \dots w_n) = p(w_1)p(w_2)p(w_3) \dots p(w_n\ )$$

The $p(w)$ and $p(d)$can be computed by observing the whole dataset. Thus we only need to maximize $p(c|d)$.

Suppose the training dataset contains m pair of the citation contexts and cited papers. Then the goal for us is to maximize the probability for all these cited papers to be cited. I.e. we maximize:

$$\prod_{t}^{m} p(c_t|d_t)$$

We apply the logarithm to it; thus we get:

$$\sum_{t}^{m} log p(c_t|d_t)$$

As we talked earlier, words in the citation context c is considered to be mutually conditional independent. Therefore, we have:

$$p(c_t|d_t) = p(w_{t_1}, w_{t_2}, w_{t_3} \dots w_{|c_t|}|d_t) = \prod_{i=1}^{|c_t|} p(w_{t_i}|d_t)$$

Therefore, the objective function for the NPM can be depicted as:

$$\sum_{t=1}^{m} \sum_{i=1}^{|c_t|} log p(w_{t_i}|d_t)$$

The NPM model uses neural networks to compute $p(w|d)$, i.e. the probability for a specific word to occur when cited document is d_t. Specifically, we have:

$$p_\theta(w|d) = \frac{exp(s_\theta(w,d))}{\sum_{i=1}^{|V|} exp(s_\theta(w_i,d))}$$

Where $\theta$is the trainable parameters for the score neural networks denoted as s_θ. This score neural networks takes word w and document d as inputs and output a score. The word w is mapped to word embedding v_w by the score neural networks. The document d is also mapped to document embedding v_d with the same dimension of v_w. Then the score neural networks compute the inner product of the document embedding v_d and word embedding v_w. After that, the score neural networks use the sigmoid function to map it to values between 0 and 1. The whole process can be depicted as:

$$s_\theta(w,d) = f({v_w}^T v_d)$$

Since the vocabulary size for a whole dataset is to large, computation for the initial $p_\theta(w|d)$ is too time consuming. Thus the NPM applies the negative sampling proposed in the skip-gram model (Mikolov et al. 2013) to accelerate the computation.

For a given word w in the citation contexts, the skip-gram model chooses words that close the given word $w$in the citation contexts as positive examples, and randomly chooses words the vocabulary sets as negative examples. Therefore, the purpose for training is to maximize the log-likelihood:

$$l_m(\theta) = log s_\theta(w_p, w_m) + \sum_{i=1}^{k} log(1 - s_\theta(w_{n_i}, w_m))$$

2.6.2.2 the Neural Citation Network

The Neural Citation Network (NCN) (Ebesu, T. et al. 2017) is another representative model for local citation recommendation. This is the first model that uses the semantic information from the citation contexts rather than bags of words. It uses an encoder-decoder architecture as well as the attention mechanism.

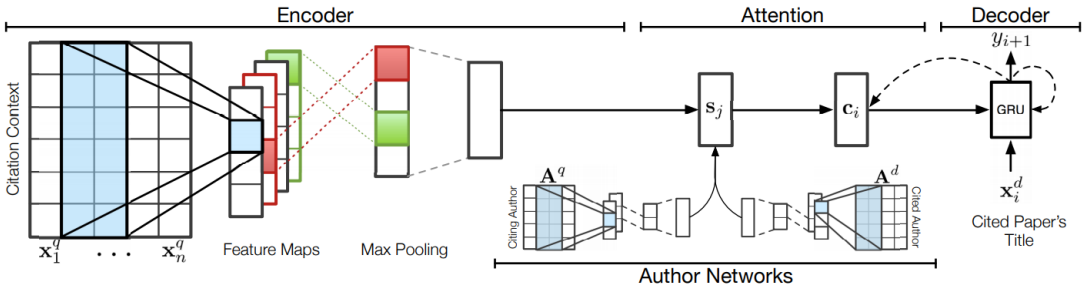

Figure 2.5: The architecture of the Neural Citation Network (NCN) with the attention mechanism and author networks.

The architecture of the NCN is depicted in Fig. 2.5. As we talked earlier, it is an encoder-decoder model. Their encoder uses the Time-Delay Neural Networks (TDNNs) (Collobert, et. al. 2008). It is a modification to the classic convolutional neural network. For a given sentence, it can be seen as word sequences. Every word can be mapped to word embeddings. Thus the given sentence can be seen as a sequence of word embeddings. The TDNN computes the convolution of the convolution kernel and word embeddings over the time axis, i.e. the sequence of words.

Specifically, for a given sentence, we suppose its length to be n. For the t^{th} word $word_t$, we assume it is $q$dimensional. Thus the matrix for the whole sentence can be seen as ${w^q}_{1:n} = {w^q}_1 \oplus {w^q}_2 \oplus \ldots \oplus {w^q}_n$, which is the concatenation of all the word embeddings along the time axis. Suppose a convolutional filter $w^{l \times q}$compute over the scope of l word embeddings at a time over the time axis. We use the rectifier (ReLU) as the activation function. Thus we have:

$$value = ReLU(w^T {x^q}_{k:k+l-1} + b_k)$$

Where $w$and $b_k$is the trainable parameters in the convolutional neural network. After that, we use the max pooling over time to get a more stable feature. Finally, the feed-forward neural network is used to extract information furtherly. It is worth noting that the filter size is not fixed. Different filter can have different scope l to get various levels of understanding for phrases, e.g. bigrams, trigrams. To capture a more comprehensive and wide range understanding of sentences, this whole process is repeated multiple times.

The TDNN can compute all these filters in parallel, which makes it much more time efficient than the RNN.

For the decoder part, the NCN uses a different model. The inputs for the decoder are cited paper titles, and the titles are usually much shorter than the citation contexts, but have more comprehensive information. Therefore, the NCN chooses to use RNN to process it due to the better capability of RNN to remember long scope previous information. Specifically, they use the Gated Recurrent Unit (GRU) (Cho, K. et al. 2014) to help prevent the gradient vanish or explosion problem.

As we talked earlier, the TCNN is much more time efficient. However, its ability to handle long span information is still limited due to its architecture. The max pooling layer also has some drawbacks. Words in the margin of the sentences get less influence in the output compared to other words. Therefore, the NCN uses the attention mechanism to adjust the weights of the output from the encoder. The attention weight is computed from the hidden stats of the RNN decoder and the outputs from the TCNN.

The NCN also uses the author names to further boost the performance. As indicated in their paper, the author names have a large impact on the actual citation. Therefore, the NCN treats the author names as context information and passes them into the TDNNs to extract feature representations too. It is worth noting that the same person may have different behaviour between citing others' papers or being cited. For example, a famous researcher's paper may be cited by a great number of people, but he can't cite that many papers while writing his own paper. Therefore, the author name of the citing papers and cited papers are mapped to different vector spaces. Then the output of the author networks is concatenated with the output of the citation contexts encoder and passed to the decoder.

For the encoder part, the NCN uses a RNN network to take the candidate paper titles as input. After that, the final output of the RNN network is passed into a softmax layer to project the output to the probability distribution over the vocabulary. Each time it only takes one word and bases on it as well as all the previous input words to give a predicted probability:

$$p(y_i|y_{\leq i}, s) = softmax(Vh_i)$$

The NCN uses the mutual probability of all the words in the candidate paper title as the measure. It computes the log sum of probability:

$$logP(y) = \sum_{i}^{m} logP(y_i|y \leq i, s)$$

3. Methodology

In this chapter, we explain our approach for solving the citation recommendation problems in detail. Firstly, we analyze the problem we aim to solve and make a clearer illustration of the citation recommendation system we hope to build, as well as the input and output of the system. Secondly, we demonstrate the architecture of our model, and the function of each built module in our model. Finally, we show our thoughts in the training process in detail.

3.1 Problem Definition

We mainly focus on the local citation recommendation problem. The local citation recommendation means that we use citation contexts information of the citing papers, rather than the whole manuscripts of the citing papers, to make recommendations. Some information from the cited papers are also needed, such as the title and abstract of the cited papers. The aim of local citation recommendations is to provide users lists of reasonable candidate papers to cite from, based on just several sentences most close to the citation location. Therefore, users can use local citation recommendation systems to get timely recommendations while writing their drafts.

3.2 Model

In order to solve the local citation recommendation problems. We build an innovative neural network model. Our model is composed of three parts: 1) Citation contexts encoder module. 2) Embedding remap module. 3) Grading module. Fig. 3.2 illustrates the simplified process in solving the local citation recommendation problems through our approach.

Firstly, we use the citation contexts encoder module to extract information from citation contexts and cited papers. The citation contexts encoder module uses a pretrained sentence embedding model. The citation contexts before and after the citation locations are passed into the citation contexts encoder module separately. The citation contexts encoder module takes these input and output embeddings contain essential information

to represent citation contexts. If additional metadata, such as titles and abstracts of citation contexts are provided, they can also be used in extracting the citation context embeddings. Metadata of cited papers (or candidate papers) are also passed into the citation contexts encoder module. The module then converses the inputs to embeddings containing essential information to represent cited papers. We add different weights to different sentences and metadata. These weights are learnable parameters. Then we concatenate them together.

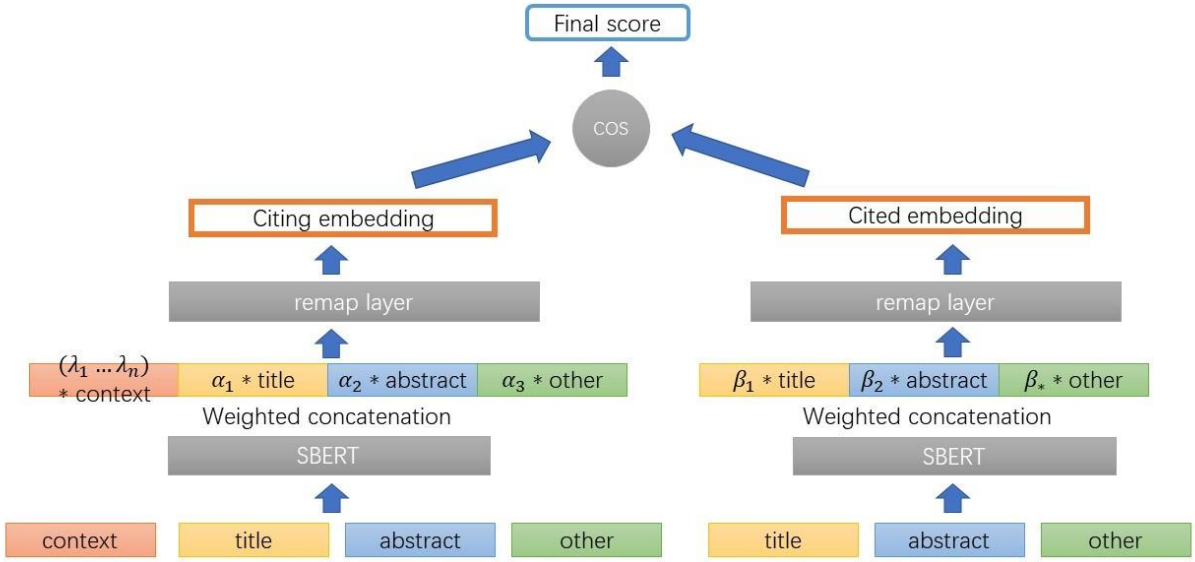

Figure 3.2 Model architecture

Secondly, since the citation contexts encoder module uses the pretrained sentence embedding model, the output embeddings for citation contexts and cited papers are not totally accurate. Moreover, the dimensions of them are also not the same. Therefore, we use the embedding remap modules to make some modifications for them. The embeddings for citations contexts and cited papers are passed into this module and remapped separately to the same vector space. Thus, we get more embeddings of the same dimensions for further processing.

Finally, the new embeddings are passed into the grading module. This module computes the cosine similarity for the citing and cited papers. The scores are between 0 and 1. The higher the scores, the more recommended the candidate papers are.

3.2.1 Citation Contexts Encoder Module

To extract semantic information from citation contexts and cited papers (or candidate papers), we use the pre-trained Sentence-BERT model (Reimers, N. et al. 2019). This model is a modification of the BERT model (Devlin, J. et al., 2018) using siamese and triplet networks. The original BERT model is aimed to provide a backbone for all kinds of natural language processing problems, which makes it not much suitable for getting accurate sentence embeddings. The Sentence-BERT model is designed to solve this problem. The BERT model's output is varied, while the Sentence-BERT model adds a pooling layer to the end of it and makes the final output to a fixed length. The Sentence-BERT model uses pairs of sentences and computes their similarity based on cosine similarity or Manhattan / Euclidean distance for training. This makes the Sentence-BERT model pretty efficient in semantic similarity search and for clustering problems.

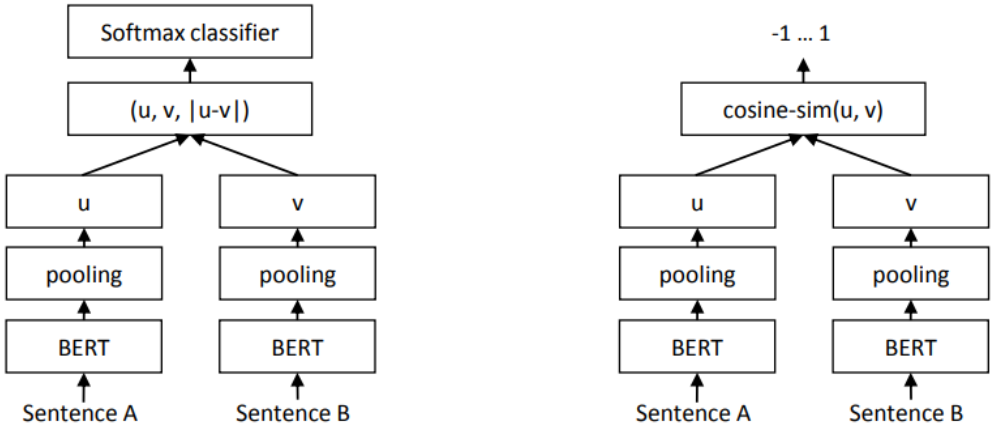

Figure 3.3 Sentence-BERT training model on classification problems (left) and similarity compute problems (right)

Fig. 3.3 depicts two training circumstances for the Sentence-BERT model. The left one is with the classification objective function. The two BERT models in it have tied weights. We input sentences A and B to BERTs separately. The outputs from BERT then get mapped in pooling layers and change to *u* and *v*. Then we compute the distance between *u* and *v,* which is denoted as $|u - v|$. After that, we concatenate *u, v* and $|u - v|$together and input it to the softmax classifier. It can be depicted as:

$$o = softmax(W(u, v, |u - v|))$$

where $W \in R^{3n \times k}$ is a trainable weight matrix. *n* is the dimension of the sentence embeddings, i.e. *u,v* and $|u - v|$. *k* is the number of labels. And the goal of this task is to optimize the cross entropy.

The right one is with the regressive objective function. This task focuses on computing the cosine similarity of the two sentence embeddings *u* and *v*. And the goal of this task is to minimize the mean square error loss between the outputs and ground truths.

The Sentence-BERT model also uses a Siamese and triplet training method. For every anchor sentence *s*, we have a positive sentence *p,* and a negative sentence *n.* Then we can get the differences between *s* and *p*/*n*, which is denoted as $|s - p|$and $|s - n|$. We train the network to make $|s - p|$significantly smaller than $|s - n|$. Mathematically, we minimize the following loss function:

$$max(0, |s - p| - |s - n| + \varepsilon)$$

The ε denotes the least margin, which means that the positive sentence *p* should be at least ε closer to *s* than the negative sentence n. And ε is set to be 1 in the Sentence-BERT model training process.

Fig. 3.4 shows the architecture of sentence encoding part of the citation contexts encoder module. Since the Sentence-BERT model (SBERT in the figure) has already been well trained in a really large corpus, i.e., the combination of the SNLI (Bowman, S. R. et al. 2015) and the Multi-Genre NLI (Williams, A. et al. 2018) dataset, and that it has too many parameters for us to train in a reasonable time, we just keep its weight parameters frozen. For citing document, we pass the left citation contexts, right citation contexts, cited paper titles and cited paper abstracts to the Sentence-BERT model separately. The citation contexts are composed of a series of sentences. Each sentence is processed by the Sentence-BERT model separately. After processed by the Sentence-BERT model, we get a series of sentence embeddings. We add weight to these sentence embeddings. Then we concatenate them together. For cited document, a similar process is used except we do not have citation contexts. Therefore, the final feature representations for citing

and cited documents have different dimensions. We handle this issue later in the remap module.

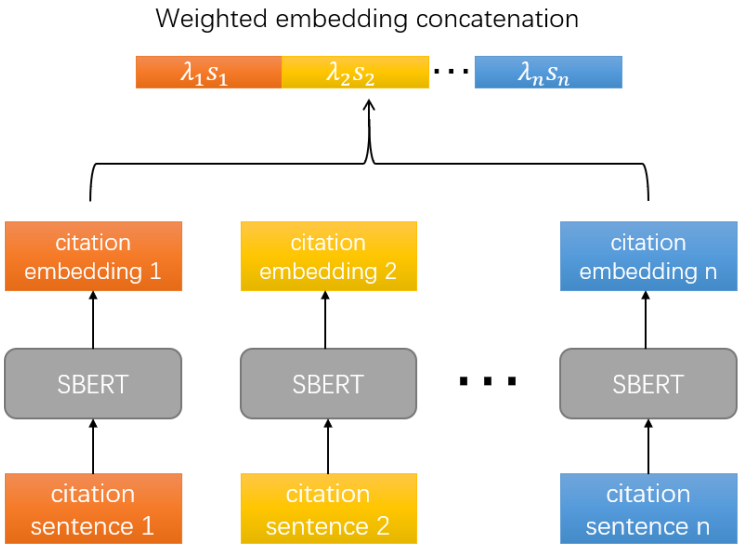

Figure 3.4 the architecture of the citation contexts encoder module.

3.2.2 Embedding Remap Module

Fig. 3.5 shows the architecture of our embedding remap module. Although the Sentence-BERT is already pretty efficient, it is designed for general sentence embedding. Thus it is not totally suitable for our specific citation contexts and cited document embedding task. Moreover, our feature representations got from the citation context encoder module have different dimensions as we mentioned before. Therefore, we design this remap module to refine the embeddings extracted by our citation contexts encoder module and map them to the same vector space.

For every citation context embedding u_1, we multiply it with the trainable weight $W_1 \in R^{m_1 \times n_1}$, it can be depicted as:

$$v_1 = u_1 \times W_1$$

Where v_1 is the new citation context embedding, m_1 is the dimension of initial citation context embedding, n_1 is the dimension of the new citation context embedding.

The same are with cited document (or candidate paper) embeddings. For every cited document embedding u_2, we multiply it with the trainable weight $W_2 \in R^{m_2 \times n_2}$, it can be depicted as:

$$v_2 = u_2 \times W_2$$

Where v_2 is the new cited document embedding, m_2 is the dimension of initial cited document embedding, n_2 is the dimension of the new cited document embedding.

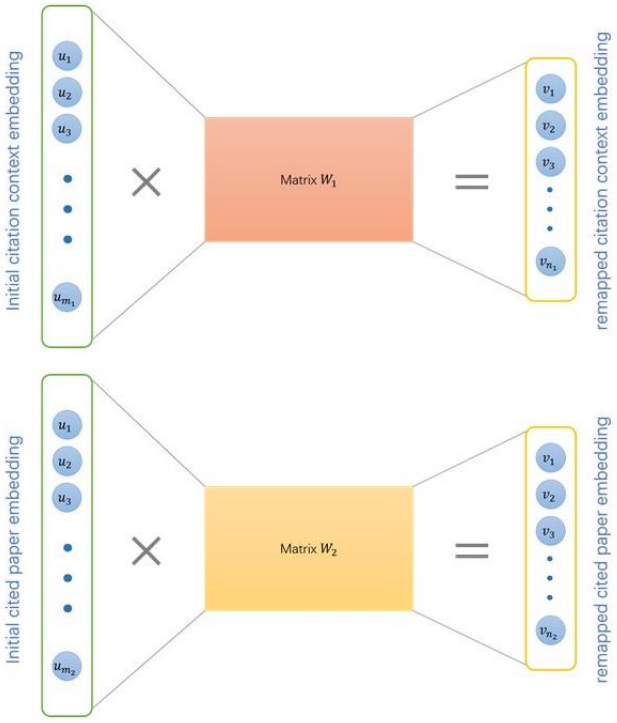

Figure 3.5 Embedding remap module

3.2.2 Grading Module

The grading module is composed of a single layer of cosine similarity. Specifically, given the embeddings for citing document as x_1, the embeddings for cited document as x_2, we have

$$similarity = \frac{x_1 \cdot x_2}{\max\left(\left|\left|x_1\right|\right|_2 \cdot \left|\left|x_2\right|\right|_2, \epsilon\right)}$$

Where ϵ is a very small value to avoid division by zero.

We use the similarity as our final score for measuring the correlation between the citing document and cited document. The score is between zero and one. The higher the score, the more possible that the cited document is actually cited by the citing document.

3.3 Learning

In this section we briefly show some core ideas in our training process.

3.3.1 Modified Negative Sampling

We use negative sampling proposed in the skip-gram model (Mikolov et al. 2013) to generate the negative examples. And we modify this approach a little bit. For every citation context, we randomly choose a candidate paper in cited paper dataset different from the ground truth as a negative example.

3.3.2 Triplet Loss

Our model is trained using a triplet set of d_p, d_q, d_q^-, where d_p is the citing paper, d_q is the cited paper as we stated before, and d_q^- is the paper randomly chosen from our cited paper dataset that is notd_q. Our model aims to predict a high score for the input pair of (d_p, d_q), and predict a relatively low score for the input pair of d_p, d_q^-. The loss function is defined using the per-instance triplet loss (Wang J. et at., 2014):

$$loss = \max\left(\eta + s\left(d_p, d_q^-\right) - s\left(d_p, d_q\right), 0\right)$$

Where $s\left(d_i, d_j\right)$ is our model output given input pair $\left(d_i, d_j\right)$, and η is a hyperparameter of our model.

3.3.3 Optimization Algorithm

We use the Adam optimization algorithm (Kingma et al. 2014), which is an extension to the stochastic gradient descent algorithm (Robbins, H. 1951), to adjust the weights in our neural network model.

Adam can be considered as a combination of Root Mean Square Propagation (RMSprop) and momentum. Similar to RMSprop, which uses exponential moving average for second-order momentum, Adam uses exponential moving average to calculate not only second-order momentum but also first-order momentum.

First, we compute the gradient towards the objective function (i.e., the loss function).

$$g_t = \nabla_\theta J(\theta)$$

Second, we calculate the first and second order momentum based on historical gradients:

$$m_t = \eta[\beta_1 m_{t-1} + (1-\beta_1) g_t]$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2).diag({g_t}^2)$$

where η is the learning rate; ${\beta}_1, {\beta}_2$is the Momentum. $m_0 = 0, v_0 = 0$. $g_t \odot g_t$is denoted as ${g_t}^2$. $diag$means diagonal matrix. It is worth noting that there is a shift to the initial value in the beginning stage of the iteration. Therefore, we can do some bias correction to the first and second order momentum:

$$\widehat{m_t} = \frac{m_t}{1-{\beta^t}_1}$$

$$\widehat{v_t} = \frac{v_t}{1-{\beta^t}_2}$$

Then we have,

$$\theta_{t+1} = \theta_t - \frac{1}{\sqrt{\widehat{v_t}} + \varepsilon}\widehat{m_t}$$

This makes the iterations more stable.

4. Experiment

4.1 Dataset

In this thesis, we use the FullTextPeerRead dataset (Jeong, Chanwoo et al. 2019) as our benchmark dataset. This dataset is created by processing the PeerRead dataset (Kang, D. et al. 2018).

Header	Description
target_id	Citing paper id
source_id	Cited paper id
left_citated_text	Text to the left of the citation tag when citing
right_citated_text	Text to the right of the citation tag when citing
target_year	Release citing paper year
source_year	Release cited paper year
target_title	Citing paper title
source_title	Cited paper title
target_abstract	Citing paper abstract
source_abstract	Cited paper abstract
target_author	Author names of citing paper
source_author	Author names of cited paper
target_venue	Name of venue citing paper published in
source_venue	Name of venue cited paper published in

Table 4.1 Description of elements in dataset

As depicted in Table 4.1, this dataset contains context information (title, abstract, citation contexts) and metadata (author names, published venues, published years) for both citing

papers and cited papers. A concrete example is shown in Fig. 4.1. The [REF] indicates that the citing paper makes a citation here. The sentences before [REF], i.e., “The feature-based approach, such as ELMo”, is called left_citated_text in the dataset. The sentences after [REF], i.e., “,uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT)”, is called right_citated_text. (It is worth mentioning that the previous sentences can also be seen as left_citated_text for the second [REF].)

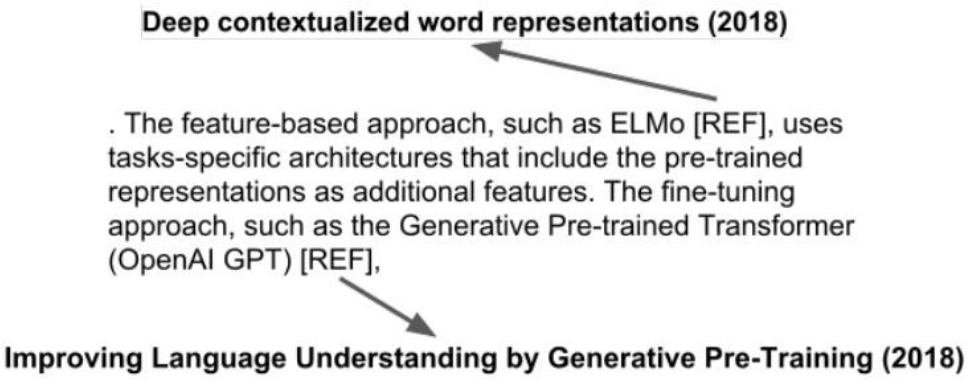

Figure 4.1: An example citation context with citation placeholders to find potential references. This example is from the BERT paper (Devlin, J. et al., 2018).

This dataset contains 4898 papers in total. 3761papers show as citing papers. 2478 papers show as cited papers. It contains altogether 17274 citation contexts. Papers in it were published between 2007 and 2017. There is some noise and errors in this dataset. For example, some words may not be parse correctly; two words may be connected and seen as one word. For these noise data, we see them as unknown token in the input. Therefore, they are just ignored by the sentence encoder.

4.2 Performance Evaluation Method

We use accuracy, Recall Top@K and Mean Reciprocal Rank (MRR) to measure our experiment performance. The accuracy is defined as the ratio for the model to make the correct prediction for whether the input is a positive or negative example. Recall Top@K is defined as the ratio of the ground truth cited paper occurring in a Top@K

recommendation list. MRR takes the rank sequences of cited papers in the recommendation list into consideration. It is defined as follows:

$$MRR = \frac{1}{I}\sum_{i \in I}\frac{1}{index(i)}$$

Where I denotes the set of all citing papers, index(i) indicates the index of the ground truth cited paper in the recommendation list of citing paper i.

4.3 Implementation Detail of Our Method

As discussed in Chapter 3, our neural network model is composed of three parts: the citation contexts encoder module, embedding remap module and the grading module.

The core part in our citation contexts encoder module is the Sentence-BERT module (Reimers, N. et al. 2019). Specifically, we use the bert-base-nli-mean-tokens pretrained Sentence-BERT model. The Sentence-BERT model accepts sentences with variable lengths and output vectors of fixed size (768). We try variable length for the left and right citation contexts. It results out that 3 sentences for the left citation contexts and 2 sentences for the right citation contexts is the best in our case. We pass each sentence to the Sentence-BERT separately. Therefore, the dimension of the citation context embeddings in our model is 3840. For the cited papers (or candidate papers), we process their abstracts as single sentences. Therefore, the dimension of the cited paper embedding is 1536. (If the titles and abstract of the citation contexts are provided, the dimension of the citation context embeddings is 5376). We add scaler parameters to each embedding of sentence as well as metadata.

For the remap module, we use the basic linear layers without activation to remap the embeddings output from the citation contexts encoder module[1]. We use two separated remap layers for the citation context embeddings and the cited paper embeddings. Since the embeddings of citation contexts and the cited papers have different dimensions, our

[1] We have tried different activation function for the remap linear previously. We also tried multi-layer neural networks. However, all of these choices make the final performance much worse. This may be due to that over complicated neural network architecture may leave out too much valuable information extracted from the initial inputs by the sentence BERT model.

remap layers also have different dimensions. The dimension for the weight matrix of the layer for citation context embeddings is 3849*768. The dimension for the weight matrix of the lay for cited paper embeddings is 1536*768.

For grading module, we use a cosine similarity layer as we mentioned before. The ϵ is set to $1 * e^{-8}$.

The values for all learnable weights of our linear layers are initialized for $U(-\sqrt{k}, \sqrt{k})$. For the Adam optimizer, we set the initial learning rate to be 0.001. We set the coefficients used for computing running averages of the gradients and its square to be 0.9 and 0.999. We don't use the weight decay.

We use mini-batch gradient descent and set the batch size for positive training examples 128. We generate the same number of negative examples randomly for every batch. We use dropout for the grading module and set the dropout probability to be 0.5.

4.4 Baselines

4.4.1 RNN & RNN model

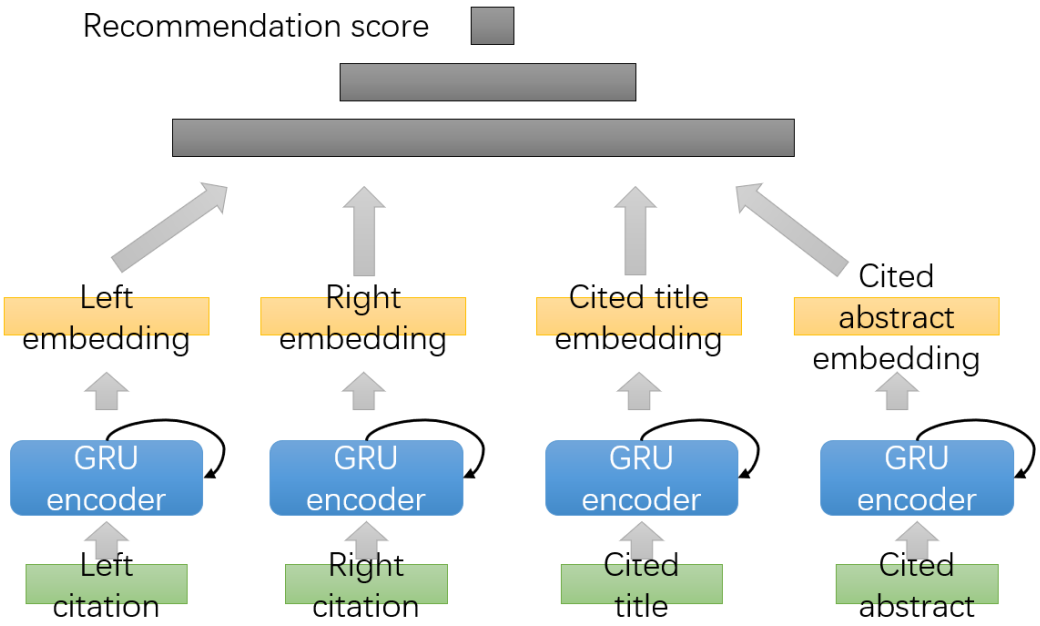

Figure 4.2 Architecture of the RNN & RNN model

The RNN & RNN model uses RNN to handle both the citation contexts and the cited papers. We use the GRU (Cho, K. et al. 2014) to encode the left citation contexts, right citation contexts, cited paper titles and cited papers abstract separately. The hidden state dimension of all the GRUs are set as 1024. We use the pre-trained word to vector embeddings from spaCy (Honnibal, M. et al. 2017), which traverses words to embeddings with the dimension of 300. We use gradient clipping (Zhang, J. et al. 2019) and set the max threshold to be 1. The structure is depicted in Fig. 4.2.

4.4.2 BERT-GCN model

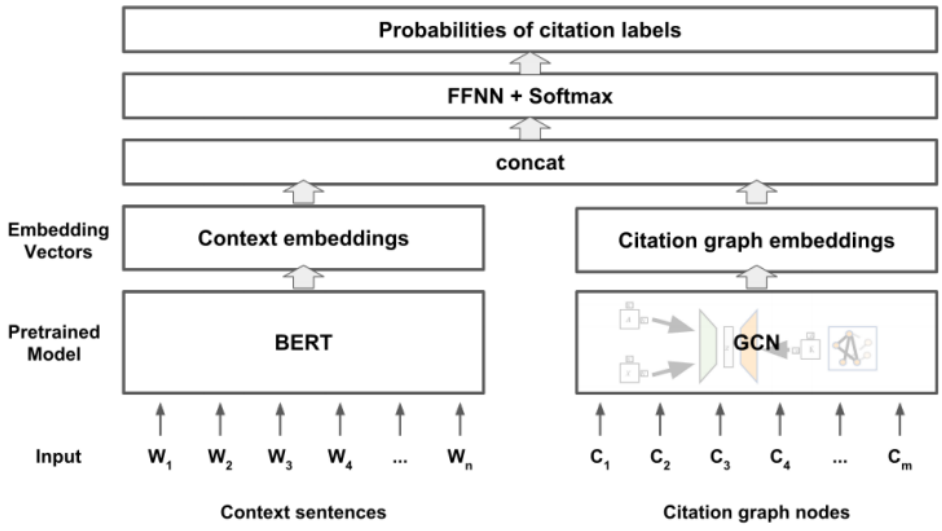

Figure 4.3 Architecture of the BERT-GCN model

The Fig. 4.3 depicts the architecture of the BERT-GCN model. It uses BERT model to process the input context sentences and output context embeddings. Moreover, it uses GCN to process the citation graph. Then it concatenates the outputs of these two modules together. Then the concatenated vector is projected into the feedforward neural networks and used to compute the final probability.

4.5 Quantitative Results

We train our model and the baselines for 400 epochs in the same benchmark dataset we mentioned before. We split the dataset randomly into training and testing dataset

according to the ratio of 8:2. This leads to 13808 citations in our training set and 3452 citations in our testing set. We get the following experiment results.

	Recall@5	Recall@10	Recall@30	Recall@50	Recall@80	MRR
RNN-to-RNN	0.31	0.38	0.51	0.57	0.64	0.23
BERT-GCN	0.49	0.53	0.60	0.65	0.70	0.42
Our model -	0.41	0.49	0.62	0.68	0.74	0.26
Our model	0.54	0.64	0.77	0.82	0.85	0.35
Our model +	0.56	0.66	0.78	0.82	0.85	0.36

Table 4.2: An overview of experiment results measured by MRR and Recall.

Table 4.2 reports the recall and MRR results for the two baselines and three variants of our method.

The first variant, labeled “Our Model -”, only uses citation contexts and titles of cited paper. Although the provided information is highly limited, it still achieves impressive performance.

The second variant, labeled “Our Model”, uses all the semantic information (i.e., citation contexts, titles and abstracts of both citing and cited paper). This boosts the performance greatly. This variant outperforms the BERT-GCN by a clear margin on recall, though a little lower on MRR. Considering that the BERT-GCN preprocesses the dataset and filters out a large number of relatively low-cited documents and requires a dense citation graph as we stated before, while our model uses all papers in the dataset and is solely content based, this outcome is quite reasonable. Moreover, since our model is solely content based, our model does not prefer papers that are highly cited or published in famous conferences. This effectively helps researchers to overcome the Google scholar effect (i.e., researchers tend to cite papers appearing at the top of the Google search results, without checking the actual relevance of these citations to their papers. Serenko, A., 2015). This content-based recommendation method also makes our model also generalizes well to new papers not presented in our dataset.

The third variant, labeled “Our Model +”, further encodes metadata (i.e., author names and venues) to our model. This furtherly improves the performance of our model.

On the other hand, when using the same training batch size, it takes about 36 minutes and 38 seconds to train the RNN-to-RNN model for 100 epochs, while our model only needs 6 minutes and 36 seconds. This outcome clearly demonstrates the advantage of our lightweight model. In practical situations, since a large number of papers are published in different conferences every year, the citation recommendation dataset may be updated from time to time. Therefore, our lightweight model, which is easy to train and update, has a much better application prospect.

4.6 Qualitative Study

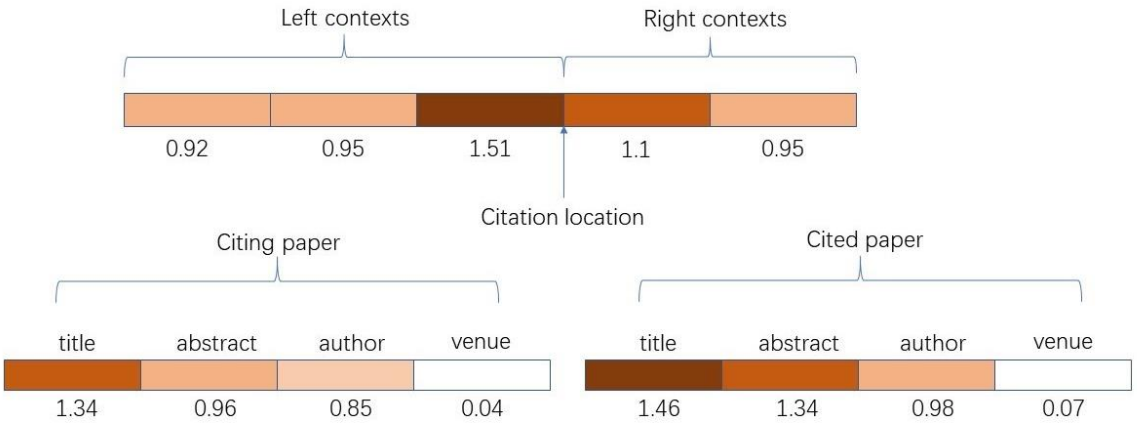

Figure 4.4: weights assigned to different parts of citation contexts and metadata. Each block for citation contexts denotes one sentence.

We analyze the weights (i.e., λ, α, β) learned during training and show them in Fig. 4.4. It is clear that our model succeeds in learning to pay different levels of attention to adequate parts of citation contexts and metadata. For citation contexts, the sentence right before the citation location raises the most attention while the sentence after the location is the second. This is consistent with human behaviors. The titles and abstracts also contribute a lot. Unlike previous study (Ebesu, T. et al., 2017), the weight value for the author

information is relatively less than the weight values for title and abstract, which may indicate that the author information is less important in our model. Finally, our model tends to omit the venue information since the weight value for it are much close to zero. This makes sense since papers in the benchmark dataset are all from top-tier venues of artificial intelligence.[2]

Table 4.3 shows two concrete examples of our model's top 3 recommendations for a specific citation context. It is worth noting that the citation contexts contain some noise. Some words in the contexts is concatenated together, such as the "asdiscussed" and the "newcitehill2016learning", which should be "as discussed" and "new citehill 2016 learning" in fact. Although there is much noise data, our model still gets reasonable recommendations, which prove our model's accuracy and robustness.

Citing title 1: Gradients of Counterfactuals
Citation context 1: ever, asdiscussed earlier, naively using the gradients at the actual input doesnot accurate quantify feature importance as gradients suffer fromsaturation.Score back-propagation based methods. The second set ofapproaches involve backpropagating the final prediction score througheach layer of the network down to the individual features. Theseinclude DeepLift , Layer-wise relevance propagation LRPder . 2016, Deconvolution networks DeConvNets & 2014, andded back-propagation
Top recommendations 1: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift Estimating or Propagating Gradients Through Stochastic Neurons Maxout Networks Network In Network

[2] We also tested using different combinations of only a part of various metadata and compare the final performances. The results we get are consistent with the conclusions drawn from our analysis of the attention mechanism.

Citing title 2: Learning Paraphrastic Sentence Embeddings from Back-Translated Bitext
Citation context 2: Other work in learning general purpose sentence embeddings has used autoencoders. encoder-decoder architectures or other learning frameworks.newcitewieting-16-full and newcitehill2016learning provide many empirical comparisons to this prior work. For conciseness, we compare only to the strongest configurations from their results.Paraphrase generation and discovery.
Top recommendations 1: Teaching Machines to Read and Comprehend Learning to Compose Neural Networks for Question Answering Pointer Networks Sequence to Sequence Learning with Neural Networks Learning Dependency-Based Compositional Semantics

Table 4.3 Two examples of the top 3 recommendations for our model.

5. Conclusion and Future Work

In this chapter we summarize our work presented in this thesis. Lastly, we discuss our future plans.

5.1 Conclusion

The local citation recommendation has growing demand in recent years. Many researchers use different methods to solve this problem. Research has shown that the traditional RNN model is slow in processing texts with really large spans, such as paragraphs. The TDNN is proposed to solve this problem. Although it computes much faster than the RNN model, its accuracy drops as the length of the sentences needed to be processed grows. This limitation makes previous research usually just use short citation context sequences. Some long context information, such as abstract which may be also valuable, are omitted due to this limitation.

In this thesis, we promote an innovative approach to solve this problem. We design a neural network model which is based on the pretrained Sentence-BERT as encoder. The pretrained model dramatically boosts the acceleration of the training process and makes it possible for extracting information in large scale texts. Since the Sentence-BERT is trained for general sentence embedding but not the specific citation recommendation task, the embeddings it generated are not totally accurate. Besides, the embeddings of citing and cited papers have different dimensions. Therefore, we use the remapping layer to refine the embeddings and map them to the same vector space. Finally, the cosine similarity is used to get the recommendation scores.

For the experimentation results, it can be observed that our model outperforms the baselines by a clear margin in most cases. And the use of additional information boosts the performance furtherly. Our model also uses much less time to train over the same amount of data. Besides, our model shows robustness against noise data, which makes our model more applicable to different circumstances.

In conclusion, we propose a powerful and efficient model to give citation recommendation. Our model mainly uses the citation context information and focuses on the semantic

analysis. Additional information can also boost our model's performance. Our model is also explainable, which makes our model more reliable. We believe this work will boost future research on the local citation recommendation area.

5.2 Future Work

We believe this work can provide a solid step towards solving the local citation recommendation problem. However, more aspects in this research area can be explored further.

First of all, we focus on research of the relationship between citation contexts and cited documents. For one citation context, we only take one cited paper into consideration. However, the previous citation may influence the following citation behavior. Further research may explore the influence of previous citations to the citation in a specific place.

Secondly, our works do not distinguish between different topics. Modern scientific research has so many different areas. Researchers in different research areas may have different ways of thinking. Thus their citation behavior may vary a lot. Further research may explore the difference citation behaviors between research areas thoroughly.

Lastly, the computation process of providing reasonable citation recommendations is really time consuming. All previous works (include us) need to extract information from all the candidate papers and compute a score for each of them using neural networks, which is really time and resource consuming. Future work may think about a more efficient way to find suitable citation recommendations.

Bibliography

Robbins, H., & Monro, S. (1951). A stochastic approximation method. The annals of mathematical statistics, 400-407.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks, 5(2), 157-166.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Collobert, R., & Weston, J. (2008, July). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning* (pp. 160-167).

Tang, J., & Zhang, J. (2009, April). A discriminative approach to topic-based citation recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 572-579). Springer, Berlin, Heidelberg.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111-3119).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26, 3111-3119.

Wang, J., Song, Y., Leung, T., Rosenberg, C., Wang, J., Philbin, J., ... & Wu, Y. (2014). Learning fine-grained image similarity with deep ranking. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1386-1393).

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.

Küçüktunç, O., Saule, E., Kaya, K., & Çatalyürek, Ü. V. (2014). Diversifying citation recommendations. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *5*(4), 1-21.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Tang, X., Wan, X., & Zhang, X. (2014, July). Cross-language context-aware citation recommendation in scientific articles. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval* (pp. 817-826).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems (pp. 3104-3112).

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Huang, W., Wu, Z., Liang, C., Mitra, P., & Giles, C. L. (2015, February). A neural probabilistic model for context based citation recommendation. In *Twenty-ninth AAAI conference on artificial intelligence*.

Liu, H., Kong, X., Bai, X., Wang, W., Bekele, T. M., & Xia, F. (2015). Context-based collaborative filtering for citation recommendation. *IEEE Access*, *3*, 1695-1703.

Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

Serenko, A., & Dumay, J. (2015). Citation classics published in Knowledge Management journals. Part II: studying research trends and discovering the Google Scholar Effect. Journal of Knowledge Management.

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). paper recommender systems: a literature survey. International Journal on Digital Libraries, 17(4), 305-338.

Cheng, J., Dong, L., & Lapata, M. (2016). Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135-146.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30, 5998-6008.

Honnibal, M., & Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.

Ebesu, T., & Fang, Y. (2017, August). Neural citation network for context-aware citation recommendation. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval* (pp. 1093-1096).

Williams, A., Nangia, N., & Bowman, S. R. (2017). A broad-coverage challenge corpus for sentence understanding through inference. arXiv preprint arXiv:1704.05426.

Britz, D., Goldie, A., Luong, M. T., & Le, Q. (2017). Massive exploration of neural machine translation architectures. arXiv preprint arXiv:1703.03906.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

Weiss, G., Goldberg, Y., & Yahav, E. (2018). On the practical computational power of finite precision RNNs for language recognition. arXiv preprint arXiv:1805.04908.

Bhagavatula, C., Feldman, S., Power, R., & Ammar, W. (2018). Content-based citation recommendation. *arXiv preprint arXiv:1802.08301*.

Kang, D., Ammar, W., Dalvi, B., van Zuylen, M., Kohlmeier, S., Hovy, E., & Schwartz, R. (2018). A dataset of peer reviews (peerread): Collection, insights and nlp applications. arXiv preprint arXiv:1804.09635.

Ayala-Gómez, F., Daróczy, B., Benczúr, A., Mathioudakis, M., & Gionis, A. (2018). Global citation recommendation using knowledge graphs. *Journal of Intelligent & Fuzzy Systems*, *34*(5), 3089-3100.

Tahamtan, I., & Bornmann, L. (2018). Core elements in the process of citing publications: Conceptual overview of the literature. Journal of Informetrics, 12(1), 203-216.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhang, J., He, T., Sra, S., & Jadbabaie, A. (2019). Why gradient clipping accelerates training: A theoretical justification for adaptivity. arXiv preprint arXiv:1905.11881.

Jeong, C., Jang, S., Shin, H., Park, E., & Choi, S. (2019). A context-aware citation recommendation model with BERT and graph convolutional networks. *arXiv preprint arXiv:1903.06464*.

Reimers, N., & Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Färber, M., & Jatowt, A. (2020). Citation Recommendation: Approaches and Datasets. *arXiv preprint arXiv:2002.06961*.

Curriculum Vitae

Name:	Juncheng Yin
Post-Secondary Education and Degrees:	University of Electronic Science and Technology of China Chengdu, Sichuan, China 2015 - 2019 B.A.
Honours and Awards:	Western Graduate Research Scholarships (WGRS) 2019 - 2020
Related Work Experience:	Teaching Assistant and Research Assistant The University of Western Ontario 2019 - 2020