POST-PROCESSING AND APPLICATIONS OF PRE-TRAINED MODELS FOR NATURAL LANGUAGE PROCESSING

YANG RUOSONG

PhD

The Hong Kong Polytechnic University

2022

The Hong Kong Polytechnic University
Department of Computing

Post-processing and Applications of Pre-trained Models for Natural Language Processing

Yang Ruosong

A thesis submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy
November 2021

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgment has been made in the text.

Signature: _____

Name of Student: Yang Ruosong

Abstract

Pre-trained models have enabled a new era in natural language processing. The first-generation pre-trained models, word embedding, aim to embed syntactic or semantic information into low-dimension and continuous word vectors. While the second-generation pre-trained models attempt to pre-train large language models and the architecture can be used to fine-tune various downstream tasks. However, word embedding models follow distributional hypothesis so that they cannot distinguish antonyms and rare words cannot learn precise representations. Pre-trained language models such as RoBERTa ignore coherence information, and text length during training is much longer than that in applications. Also, training pre-trained models requires powerful hardware. To tackle these issues effectively, we propose to utilize post-processing to enhance two types of pre-trained models. Besides, we also utilize two types of pre-trained models to enhance specific applications on text assessment.

In this thesis, we review existing pre-trained models as well as works about text assessment first. Then we conduct four works including two works that post-processing pre-trained models and two applications on text assessment. More specifically, in the first work, we explore how to utilize the glossary to enhance word embeddings so that the post-processed word embeddings can both capture syntactic and semantic information better. In the second work, we utilize pre-trained word embedding to solve automated post scoring. To better integrate given topics and quoted posts in forums, we propose a representation model and a matching model. In the third

work, we propose to utilize self-supervised intermediate tasks to enhance pre-trained language models. Meanwhile, we investigate how these intermediate tasks benefit downstream tasks. In the last work, we use pre-trained language models to learn text representations and proposed to combine regression loss and ranking loss to enhance the performance of automated text scoring. In addition, we conclude our work and addressed future directions.

Publications Arising from the Thesis

1. Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Youzheng Wu, Xiaodong He, "Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking", in *Findings of the 2020 Conference on Empirical Methods in Natural Language Processing* (2020).

2. Ruosong Yang, Jiannong Cao, Zhiyuan Wen, "GGP: Glossary Guided Post-processing for Word Embedding Learning", in *12th Edition of its Language Resources and Evaluation Conference* (2020).

3. Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Jiaxing Shen, "Automated Post Scoring: Evaluating Posts with Extra Topics and Quoted Posts in Online Forum", *World Wide Web Journal* (2022).

4. Ruosong Yang, Jiannong Cao, Zhiyuan Wen, Shuaiqi Liu, "Enhancing Pre-trained Models with Self-supervised Intermediate Tasks for Natural Language Understanding", manuscript aims to submit to *TACL*.

5. Zhiyuan Wen, Jiannong Cao, Ruosong Yang, Shuaiqi Liu, Jiaxing Shen, "Automatically Select Emotion for Response via Personality-affected Emotion Transition", in *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on*

Natural Language Processing: Findings (2021).

6. Shuaiqi Liu, Jiannong Cao, Ruosong Yang, Zhiyuan Wen, "Highlight-Transformer: Leveraging Key Phrase Aware Attention to Improve Abstractive Multi-Document Summarization", in *The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: Findings* (2021).

7. Zhiyuan Wen, Jiannong Cao, Ruosong Yang, Senzhang Wang, "Decode with Template: Content Preserving Sentiment Transfer", in *12th Edition of its Language Resources and Evaluation Conference* (2020).

8. Yu Yang, Jiannong Cao, Jiaxing Shen, Ruosong Yang, Zhiyuan Wen, "Learning Analytics based on Multilayer Behavior Fusion", in *13th Internation Conference on Blended Learning* (2020).

9. Shuaiqi Liu, Jiannong Cao, Ruosong Yang, Zhiyuan Wen, "Key Phrase Aware Transformer for Abstractive Summarization", *Journal of Information Processing and Management* (2022).

10. Shan Jiang, Jiannong Cao, Hanqing Wu, Ruosong Yang, Yanni Yang, "Dynamic Ring Signature: Achieving Provable Anonymity in Blockchain-based E-voting", manuscript submitted to *IEEE Transactions on Dependable and Secure Computing*.

11. Hanqing Wu, Jiannong Cao, Shan Jiang, Ruosong Yang, Yanni Yang, Jianfei He, "TSAR: a fully-distributed Trustless data ShARing platform", in *The Third IEEE Workshop on Smart Service Systems* (2018).

Acknowledgments

I am too overconfident when I get my master degree. So I easily make a decision to study for a Ph.D. The hard days in PolyU are really beyond my imagination. During these years, I deeply understood the "inner peace" that appeared in Kung Fu Panda 2. Simultaneously, I also learned a lot from Prof. Cao and our group mates.

First of all, I appreciate Prof. Cao's supervision. It is my luck to be one of your students, and I have learned a lot of methodology from you. From a fresh beginner, I studied how to select a topic, how to define a problem, how to propose a solution, and how to tell a good story. With your guidance, I also had a deep thought of writing. All these skills not knowledge have benefited me until the future.

Besides, I also thank all senior group mates including Wengen Li who helped me to register my study in PolyU, and gave me lots of suggestions no matter in study or daily life, Linchuan Xu who recommended lots of academic materials since our directions were similar, Yuqi Wang, Senzhang Wang, and Jiaxing Shen who helped me to revise my paper. Qiang Li and Hui Li supervised by other professors also taught me a lot in the study and daily life. Meanwhile, I express my thanks to Xiulong Liu, Zhuo Li, Jia Wang, Yu Yang, Yanni Yang, and Shan Jiang, who brought me so much happy time during these five years. I want to stress my thanks to Zhiyuan Wen and Shuaiqi Liu, we are the NLP group, and we contributed together to many tasks assigned by Prof. Cao. Without your help, it is hard for me to imagine how to finish these projects. In addition, thanks should be also listened to by Xiaoyin Li, Ken Lai, Minjin Zhang,

Qianyi Chen, Zhixuan Liang, Jinlin Chen, etc.

Here, I would like to thank my parents, my grandmother, uncles, and aunts, your support and understanding encouraged me to finish my Ph.D. study. Same thanks to all my friends in JDAI, and daily life.

Finally, I am grateful to Angie Chiu for your wonderful TV series. No matter your masterpiece filmed in Taiwan such as the Legend of the White Snake, Moment in Peking, and Legendary Chien Lung, or that filmed in Hong Kong including The Bund, Chor Lau Heung, etc. brought me lots of happiness and thinking. Your experience also inspired me a lot for your perfect balance of work and life as well as your proper choices at different ages. In general, you are my favorite actress. You are the light when I am in the dark. Your TV series and your experience encouraged me to overcome various difficulties and adjust my mentality.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Natural Language Processing (NLP) is an area of Artificial Intelligence (AI) that explores how machines can understand and manipulate natural language text [18]. There are lots of NLP tasks such as natural language understanding tasks including text classification, text similarity tasks, text inference tasks, question answering, etc., and natural language generation involving machine translation, text summarization, dialog system, etc. To solve these tasks, corpora or datasets are necessary. With the development of the Internet, it is easier to collect a larger volume of corpus, and construct larger datasets. Statistic models become possible since statistical probability will be more accurate with larger datasets.

Statistical NLP methods usually heavily rely on discrete handcrafted features. To get practical features, feature engineering is an important but difficult step for statistical models. Neural methods usually use low-dimensional and continuous vectors to implicitly represent the syntactic and semantic features of the natural language text. Representations for specific tasks can be easily learned during solving these tasks automatically. In summary, neural models can learn features rather than manually

designing features, therefore, it is easy for researchers to design various neural models to solve various NLP tasks.

Researchers have designed several neural networks with special architecture such as Recurrent Neural Networks (RNNs) [62, 40, 16], Convolutional Neural Networks (CNNs) [50, 47, 45, 37], Graph Neural Networks (GNNs) [48, 99, 83], Transformer [98, 49], attention mechanisms [80], and memory network [107, 88]. Different types of neural networks have the ability to capture different information from natural language texts. All these models face the same critical challenge, data hungry. Since deep neural networks usually contain a large number of parameters compared with statistical models, they are thus easy to overfit and have poor generalization ability [3, 112] without sufficient training data.

The key difficulty to obtain a large amount of training data is that labeling data is expensive, time-consuming, and labour-intensive. For example, segmenting images via crowdsourcing costs about $6.4 per image [59]. To tackle the challenge, self-supervised learning [59] which labeling the training data by leveraging the relations between different parts of input is an effective approach. Inspired by this idea, large unlabelled corpora can be possibly used to train neural models. To make full use of these data, larger models with much more parameters are necessary. Researchers attempt to stack multi-layer transformer models, and adopt Pre-LM (Pre-Layer Normalization) [74] or Post-LM [22] to alleviate gradient vanishing and explosion. All these models are called Pre-trained Models (PTM).

More generally, there are two types of pre-trained models including pre-trained word embeddings and pre-trained language models. The first-generation PTMs are pre-trained word embeddings that aim to learn low dimensional, continuous word vectors, such as Word2vec [63], and GloVe [70]. These models follow Distributional Hypothesis [35] which refers to words with similar context words contain similar semantic meaning. However, some words have several senses and show different senses in different contexts. Word embedding models only learn one representation for each word,

so they cannot tackle the multi-sense issue. These models are also called context-free word embeddings. The second-generation pre-trained models are pre-trained language models (PTLM) which focus on learning contextual word embeddings such as GPT [74], and BERT [22]. These models attempt to learn model architectures that are better semantic composition functions of words. With pre-trained language models, users don't need to design specific neural networks for specific tasks and train the models from scratch.

1.2 Motivation

For pre-trained word embedding models, they assume words with similar context words should learn similar vectors [63]. Antonyms may also occur with similar context words, but contain opposite semantic meanings [66]. Meanwhile, these models utilize the distribution of the context words of the target word to represent the semantic meaning of the target word. In practice, the volume of the corpus will affect the distribution as well as the representation quality. Also, low-frequency words lack enough context words so that they usually cannot learn accurate representations.

As for applications of pre-trained embeddings, specific tasks are required for specific semantic information. Only the pre-trained word embeddings are not enough. For example, sentiment classification and news classification need different semantic representations. So to apply word embeddings to specific applications, researchers also need to design specific neural networks for specific tasks. Word embeddings can be only used to initialize the embedding matrix in neural networks.

For pre-trained language models, RoBERTa [22] proves that the coherence task used in pre-training cannot improve the performance of downstream tasks. It also gains better performance on downstream tasks than other pre-trained models including BERT [22], StructBERT [103] that utilized various coherence tasks. However, RoBERTa only

models masked language model and misses coherence information which is important for multi-sentence tasks. All pre-trained language models are trained on general corpora such as English Wikipedia [1]. Downstream tasks come from various domains, so there is a gap between training corpora and that of downstream tasks. In practice, the sequence used in training is much longer than that in downstream tasks. The mismatch will also hurt the performance.

Exploring how to utilize pre-trained language models is also a hot topic. Since pre-trained language models are only encoder models that encode input sentences into vector representations. Existing researches only provide simple objectives for each type of task. How to design suitable objective functions for specific tasks is also a research problem. Moreover, for all pre-trained language models, the representation of the special token is used as the representation of the input sequence. Designing approaches to obtain better representations is also necessary. In addition, existing models only learn unique representations for input sequences, how to learn multiple representations for inputs with multiple sentences is still an issue.

In summary, two types of pre-trained models are still facing many drawbacks. So it is necessary to updated these models to tackle these drawbacks. In general, there are two ways. The former incorporates extra data, designs new models, and auxiliary objective functions. However, it is time-consuming and requires high-performance hardware. The later integrates extra data and designs new objective functions to post-process these pre-trained models. Since it doesn't need to re-train these models from scratch, and the volume of the extra data is much smaller than that of the training data. Post-processing is much more efficient. We mainly focus on the research of post-processing. As aforementioned, applying pre-trained models to specific tasks is not enough, we need to updates these pre-trained models with specific approaches for them. We also address the difference between post-processing and applications. For post-processing, we aim to learn better general models so that they can gain better

[1] https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

performance on various downstream tasks. For applications, we only focus on the specific task, and updated models to gain high performance on the specific task.

1.3 Research Challenges

No matter for post-processing or applications, there are three aspects that are required to be considered including data selection, neural network designing, and objective function designing. For data selection, post-processing pre-trained models need to determine the extra data first, since the selected data determine the designing of neural networks as well as objective functions. For specific tasks, we only consider how to better use the pre-trained models, we don't need extra data. So how to design proper neural networks and objective functions are two key challenges.

Before introducing details of these two challenges, I will show extra data that can be used to post-process pre-trained models first. For pre-trained word embeddings, there are usually two types of data. The first type of data is word pairs including various word pairs extracting from lexical knowledge such as synonyms, antonyms from WordNet [2], entity pairs from knowledge graphs. The other type of data is word sequences, for example, the dictionary explanation. For pre-trained language models, open-domain corpus and domain-specific corpus are two widely used extra data. Open-domain corpus means the corpus comes from various domains such as English Wikipedia[1]. And domain-specific corpus is from a specific domain, for example, the datasets of specific tasks.

The first challenge is how to design neural networks (with determined extra data). For post-processing pre-trained word embeddings, word pairs and word sequences are totally different data format, so they need to design totally different models. For applications of pre-trained word embeddings, the embeddings only provide initializa-

[2]https://wordnet.princeton.edu/

tion of word vectors, researchers are required to design specific neural networks for specific tasks. As for pre-trained language models, they contain a large number of parameters, so complicated extra neural networks will lead to convergence problems. No matter for post-processing or applications, we only utilize a simple linear layer as the extra neural network.

The second challenge is how to design suitable objective functions. For post-processing pre-trained word embeddings, different word pairs illustrate different relationships. For example, compared with antonyms, synonyms should be more similar. Different relationships should use different objective functions. Moreover, with several types of word pairs, how to tackle multiple relationships is also challenging. For applications of word embeddings, the objective functions are highly dependent on specific tasks. For post-processing pre-trained language models, designing or selecting proper tasks is significant. Different tasks are corresponding to different objective functions. For applications of PTLM, sentence representations learned from PTLM usually are enough. To further improve the performance of specific tasks, designing more complicated objective functions is more feasible.

I also address the difference between applications for pre-trained word embeddings and language models. Word embeddings are only the input of the model, so the whole neural network including encoders and other parts should be designed according to specific tasks. PTLMs have provided an encoder model, so we usually only need to design the objective functions.

1.4 Research Framework

In this thesis, I mainly focus on the post-processing of two-generation pre-trained models and two applications of text assessment.

As shown in Figure 1.1, the aim of this thesis is to learn general models and spe-

cific models based on pre-trained models including pre-trained word embedding and language models. Based on pre-trained word embeddings, both general model and specific model are required to design new architectures and objectives. While based on pre-trained language models, only task-specific objectives are necessary for both general model and specific model. In summary my research works include two general models and two specific models. All these four works will be introduced respectively.

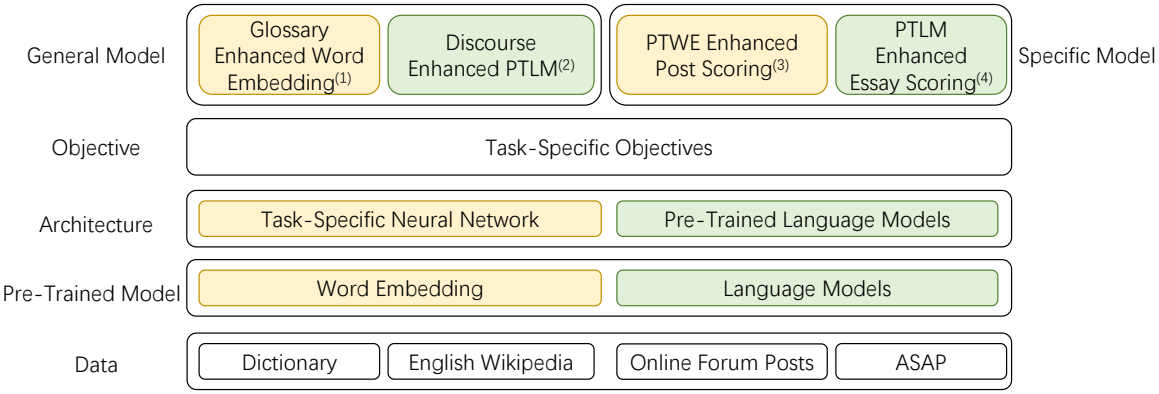


Figure 1.1: Research Framework

1.5 Thesis Organization

The rest of this thesis mainly includes a comprehensive literature review, the four tasks namely two post-processing tasks and two text assessment tasks, and future directions. For the four tasks, we introduce them in the order of enhancing word embeddings with the dictionary, applying word embedding to post scoring, enhancing pre-trained language models with coherence tasks, and applying pre-trained language models to essay scoring.

Specifically, the rest chapters of the thesis are organized as follows:

- In Chapter 2, we present a comprehensive review of pre-trained models and

various enhancement works. For word embedding, we introduce several word embedding models first. Then, there are two types of works to enhance word embedding models including retraining and post-processing. For each type of work, we illustrate existing works according to used extra knowledge such as lexical knowledge, semantic knowledge, knowledge graph. For pre-trained language models, we also introduce various pre-trained language models first. To enhance these models, intermediate tasks are widely used. We classify fine-tuned models according to whether they are general models or not. And we use this taxonomy to show existing works. In addition, how to utilize pre-trained language models on downstream tasks is also a hot topic, we also introduce various approaches that applying PTLM to specific tasks.

- In Chapter 3, we present our first work that utilizing word definitions to enhance pre-trained word embeddings. Existing post-processing models mostly consider semantic knowledge so that learned embedding models show less functional information. Compared with semantic knowledge sources, the glossary is a comprehensive linguistic resource that contains complete semantics. The previous glossary based post-processing method only process words that occurred in the glossary, and did not distinguish multiple senses of each word. To make better use of the glossary, we utilize the attention mechanism to integrate multiple sense representations. By measuring the similarity between word representation and combined sense representation, we aim to capture more topical and functional information.

- In Chapter 4, we introduce our second work about how to utilize word embedding to assess students' posts. Since the designed neural network is more significant to solve the task, we mainly introduce the designed model. Different from existing text assessment tasks, we propose a novel task, Automated Post Scoring (APS), which grading all online discussion posts in each thread of each student with given topics and quoted posts. APS evaluates not only the writing

quality of posts automatically but also the relevance to topics. To measure the relevance, we model the semantic consistency between posts and topics. We also utilize supporting arguments extracted from quoted posts to enhance posts evaluation. Specifically, we propose a mixture model including a hierarchical text model to measure the writing quality, a semantic matching model to model topic relevance, and a semantic representation model to integrate quoted posts.

- In Chapter 5, we show our third work which investigating how intermediate tasks can benefit downstream tasks. Most existing works only investigate supervised intermediate tasks. The study on self-supervised intermediate tasks is limited to the masked language model (MLM) task on specific domains. We extend the study to the general domain and other self-supervised intermediate tasks. We identify what and how these tasks can benefit general tasks without requiring extra corpora. More specifically, we identify these tasks inspired by pre-training tasks of existing PTMs. Then, we construct datasets for these tasks and fine-tune them on PTMs. To conduct a fair comparison of the identified tasks, we select RoBERTa which was trained only on the MLM task for our study.

- In Chapter 6, we illustrate our fourth work that utilizing pre-trained language models to enhance automated essay scoring. To solve the AES task, previous works utilize shallow neural networks to learn essay representations and constrain calculated scores with regression loss or ranking loss, respectively. Since shallow neural networks trained on limited samples show poor performance to capture deep semantic of texts. And without an accurate scoring function, ranking loss and regression loss measures two different aspects of the calculated scores. To improve AES's performance, we propose to fine-tune pre-trained language models with multiple losses of the same task.

- In Chapter 7, we present open challenges and future directions for post-processing

pre-trained language models. Open challenges include how to fine-tune PTLM robustly, how to alleviate forgetting learned knowledge, and how to reduce the effect of hyper-parameters. There are also several directions. In specific domains such as academic papers, patents, or clinic texts, researchers still utilize the general pre-trained language models. However, texts from different domains follow different writing guidelines and contain different characteristics. To learn knowledge from these domain-specific corpora, it is necessary to design new objective functions to capture their characteristic.

Chapter 2

Literature Review

In this chapter, we will show a brief review of existing pre-trained models, models enhancing PTMs, and existing works in text assessment. In Section 2.1, we will introduce several typical word embedding models first such as word2vec and GloVe. Then two ways to enhancing word embedding models including re-training and post-processing are introduced. For each way, we will show these models according to used knowledge, for example, lexical knowledge, semantic knowledge, and knowledge graph. In Section 2.2, We introduce various pre-trained language models including GPT, RoBERTa, etc. Also, we classify the enhanced pre-trained language models into general models and domain-specific models. Each type of enhanced model will show more details. In Section 2.3, we mainly introduce existing works in automated text/essay scoring and automatic short answer grading task.

2.1 Pre-trained Word Embedding Models

In this section, we will focus on existing word embedding models as well as various enhanced models. Specifically, we will introduce typical word embedding models in section 2.1.1. Then we introduce two types of models enhancing word embeddings

with various knowledge respectively in section 2.1.2 and section 2.1.3.

2.1.1 Word Embedding Models

Unlike signals of speech and images which have specific physical meanings, words are symbols so that the coding of words in the computer contains no meanings of words. To support various natural language tasks, it is essential to learn high-quality word representations. To qualitatively analyze the quality of word representations, researchers assume that the distance between similar words should be shorter. Researchers proposed several distributed word embedding models referring to the Distributional Hypothesis [35] that words with similar distributions of context words have similar meanings. Word2vec and GloVe are two typical models that aim to learn low dimensional and continuous word vectors, we will briefly introduce them respectively.

Word2vec toolkit[1] was released by Google in 2013. The toolkit contains two models namely Continuous Bag-Of-Words (CBOW) and Skip-Gram with Negative Sampling (SGNS). Both models can learn word representations efficiently from a large corpus. CBOW assumed that the meaning of the target word can be learned from its context words so that the model predicts the target word given its context words. Skip-Gram with Negative Sampling, on the contrary, attempts to learn the representations that can predict each context word given a target word. In the rest of this section, we will show more details of these two models.

As shown before, CBOW predicts the center word given a window of context words. Formally, CBOW predicts the word W_i according to its $2l$ contexts words as

$$P(w_i|w_{j(|j-i|\leq l,j\neq i)}) = \text{Softmax}(\boldsymbol{M}(\sum_{|j-i|\leq l,j\neq i} w_j)) \tag{2.1}$$

where $P(w_i|w_{j(|j-i|\leq l,j\neq i)})$ is the probability of word w_i predicted by its contexts, $\boldsymbol{M}$

[1] https://code.google.com/archive/p/word2vec/

is the trainable weight matrix in $\mathbb{R}^{|V|\times m}$, $|V|$ is the vocabulary size, and m is the embedding size.

CBOW minimized the sum of negative log probabilities to learn the parameters as shown in Formula 2.2.

$$\min -\sum_{i} \log P(w_i|w_{j(|j-i|\leq l, j\neq i)}) \tag{2.2}$$

SGNS utilized the target word w_i to predict various context words w_j as shown in Formula 2.3.

$$P(w_j|w_i) = \text{Softmax}(\boldsymbol{M}w_i)(|j-i| \leq l, j \neq i) \tag{2.3}$$

Similar to CBOW, $\boldsymbol{M}$ is the trainable parameters. So as the loss function as shown in Formula 2.4.

$$\min -\sum_{i} \sum_{j(|j-i|\leq l, j\neq i)} P(w_j|w_i) \tag{2.4}$$

The calculation of Softmax is highly dependent on the size of the vocabulary. SGNS is trained on a large corpus, so time efficiency is the key problem. To accelerate training, negative sampling is an effective approach. It directly samples k words as negative samples based on their word frequency. Then, it computes σ over each $k+1$ word to predict whether the word is the context word or negative sampled word.

Global Vectors for Word Representation (GloVe) [70] is another successful word embedding model. Word2vec ignored the statistics of the corpus since they focus on separate local context windows rather than global co-occurrence counts. The authors proposed a specific weighted least squares model that trained on counts of global word-word co-occurrence.

More specifically, GloVe models

$$F(w_i, w_j, \widetilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.5}$$

where $\widetilde{w} \in \mathtt{R}^d$ represents context word vectors, and P_{ij} is the probability of word j to be in the context of word i , formally

$$P_{ij} = \frac{N_{ij}}{N_i} \tag{2.6}$$

where N_{ij} is the number of co-occurrences of word j and word i in the same window, and $N_i = \sum_k N_{ik}$ is the number of times any word occurs in the context of word j.

More specifically, the dot product is adopted to model the relationship between w_i, w_j, and $\widetilde{w}_k$, Formula 2.7 is obtained.

$$F((w_i - w_j)^T \widetilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.7}$$

GloVe utilized $F = \exp$ as the solution. Then

$${w_i}^T \widetilde{w}_k = \log N_{ik} - \log N_i \tag{2.8}$$

To keep exchange symmetry, $\log N_i$ is eliminated by adding biases b_i and $\widetilde{b}_k$. The model becomes

$${w_i}^T \widetilde{w}_k + b_i + \widetilde{b}_k = \log N_{ik} \tag{2.9}$$

The loss function is defined as:

$$L = \sum_{i,j=1}^{|v|} f(N_{ij})({w_i}^T \widetilde{w}_j + b_i + \widetilde{b}_j - \log N_{ij}) \tag{2.10}$$

where $f(\cdot)$ is a weighted function.

2.1.2 Re-training Word Embedding Models

In this section, we will introduce various works that utilize lexical knowledge, categorical knowledge, relational knowledge, and knowledge graph to enhance word embed-

ding models via re-training existing word embedding models with auxiliary objective functions.

Lexical knowledge, also named morphological knowledge, are basic elements of a word such as syllables, roots, or affix (prefix and suffix). It can facilitate to identify semantically related words (e.g. words with the same root). Bian et al. [5] proposed to utilize context words to predict root, affix, and syllable of target words. Furthermore, they also use syntactic and semantic knowledge to enhance word embedding models.

Categorical knowledge includes syntactic and semantic properties of words. The syntactic category consists of its part of speech tag, syntactic role, etc. And the semantic category of a word involves its concept, semantic type, and semantic role. Levy and Goldberg [56] utilized the result of the dependency parse-trees , and redefined the context of the target word called syntactic context to train the word vectors based on skip gram model. RCM [115] mainly considered synonyms, and used the target word to predict its synonyms. RC-NET [110] extracted word pairs from the same category in Freebase [7] and assumed these two words to learn similar representations.

Relational Knowledge referred to word pairs with their relationship which is usually represented as a triplet: (Head H, Relation R, Tail T). RC-NET [110] also referred to the idea of TransE [8], and transformed the dataset about categorical knowledge into relational knowledge. SWE [58] proposed to utilize ordinal knowledge constraints such as synonym antonym rule, semantic category rule, and semantic hierarchy rule to enhance word embeddings.

The knowledge graph is a rich source of high quality, human-curated structured knowledge. Wang et al. [105] extract structured data from Freebase, and adopted a similar approach as relational knowledge to enhance word embeddings.

2.1.3 Post-processing Word Embedding Models

Faruqui et al. [28] proposed a graph-based learning approach for using semantic lexicons from PPDB [30], WordNet [64], and FrameNet [2] to post-process word embeddings, which was called "retrofitting." Jo and Choi [43] proposed extrofitting to expand more dimensions on all word vectors via filling with transferred semantic knowledge and project the vector space using Linear Discriminant Analysis. The Dict2vec [96] model constructed word pairs from both the corpus and dictionary entries. Especially for negative sampling, it ignored the words in pairs from the second part. Then Skip Gram (Word2Vec) is used to learn the word embedding model. CPAE [9] proposed a LSTM based auto-encoding model to learn word representations from dictionary definitions which are assumed to be similar to their embedding.

2.2 Pre-trained Language Models

In this section, we will introduce basic pre-trained language models in Section 2.2.1. Then various approaches that utilizing intermediate tasks to enhance pre-trained language models are shown in Section 2.2.2.

2.2.1 Pre-trained Language Models

Following the taxonomy shown in [111], we introduce two typical pre-trained language models namely GPT and BERT first. Then various post-BERT models are shown. Models combining autoregressive and autoencoding modeling are also illustrated.

GPT was the first model that combines the modern Transformer architecture and the self-supervised pre-training objective. Practically, GPT achieved significant success on almost all NLP tasks, including sentence classification, sentence similarity tasks, sentence inference tasks, question answering, and commonsense reasoning. Given

large-scale corpora without annotated labels, GPT attempt to optimize a standard autoregressive language modeling, namely, maximizing the conditional probabilities of all the words given their previous words as contexts. Formally, given a corpus consisting of token sequences $\chi = x_0, x_1, ..., x_n, x_{n+1}$, GPT applied a standard language modeling objective function by maximizing the following log-likelihood:

$$L(\chi) = \sum_{i=1}^{n+1} \log P(x_i | x_{i-k}, ..., x_{i-1}; \Theta) \tag{2.11}$$

where k is the uni-directional window size, the probability P is modeled by the Transformer decoder with parameters Θ, x_0 is the special token [CLS] which means the start of the sentence, x_{n+1} is the special token [SEP] which means the end of the single sentence or the separation of two sentences.

Unlike GPT, BERT utilized a bidirectional deep Transformer as its structure. BERT applied autoencoding language modeling rather than autoregressive language modeling that was used in GPT. Specifically, inspired by the cloze task [94], masked language modeling (MLM) is proposed as the objective function. In MLM, tokens are randomly masked with a special token [MASK], the objective of this task is to predict masked words according to contexts. Besides MLM, next sentence prediction (NSP) is also adopted as the auxiliary objective to capture coherence information between sentences. NSP adopted a binary classifier to predict whether two sentences are adjacent.

With two basic models, various post-BERT models that updated BERT to gain better performance will be introduced.

RoBERTa [60] was one of the success variants of BERT, which mainly made four simple and effective changes: (1) Removing the next sentence prediction task; (2) More training steps (epochs), with larger batch size and more corpora; (3) Longer training sentence segments; (4) Dynamically masking methods which means to mask different words in different epoch. RoBERTa outperformed BERT by a large margin on various downstream tasks. Moreover, RoBERTa also pointed out that the next

sentence prediction task is relatively useless during training RoBERTa.

ALBERT [53] was also a significant variant of BERT, which provided several effective tricks on reducing parameters. The first trick was to factorize the input word embedding matrix into two smaller ones. Then it proposed to share parameters between all Transformer layers to significantly reduce parameters. Third, it utilized a new pre-training task, called the sentence order prediction (SOP) task, to substitute BERT's NSP task.

SpanBERT [44] extended BERT by (1) changing the masking method which masking contiguous random spans rather than random tokens, and (2) proposing new pre-training tasks that utilizing the representations of span boundary to predict the entire content of the masked span, without relying on the within representations of individual tokens.

StructBERT [102] aimed to incorporate language structures into pre-training, so it proposed two new objective functions. The first objective attempted to endows the model to reconstruct the right order of intentionally shuffled word tokens. The other objective extended the sentence prediction task by predicting both the previous sentence and the next sentence.

We also show several models that combining autoregressive and autoencoding modeling.

XLNet [113] proposed the permutated language modeling to combine autoregressive and autoencoding. To avoid the shortage that the [mask] token won't appear in downstream tasks, XLNet permutated tokens' order during the pre-training and then adopting the autoregressive prediction paradigm, which helps XLNet to gain the ability of both understanding and generation.

MPNet [87] amended the XLNet's discrepancy that in pre-training XLNet does not know the sentence's length while in downstream it knows.

UniLM [25] proposed to jointly optimize different language modeling objectives together, namely unidirectional language model, bidirectional language model, and seq2seq objectives.

Recently, GLM [26] proposed a more elegant way to combine autoregressive and autoencoding. Given a variable-length masked span, Transformer blocks were required to autoregressively generate the masked tokens, instead of providing the number of [MASK] tokens to model as BERT and SpanBERT [44] done,

2.2.2 Intermediate Tasks to Enhance PTLM

To enhance pre-trained language models, an effective approach is to further train PTMs on intermediate tasks before fine-tuning them on downstream tasks. Most existing works investigated the effectiveness of supervised intermediate tasks, and some works explored how the task of the masked language model adapted PTMs to specific domains.

For supervised intermediate tasks, some researchers investigated how they benefit general tasks. Yada et al. [73] performed a large-scale study on the RoBERTa model with 110 intermediate-target task combinations to investigate when and why intermediate-task training is beneficial for a given target task. Wang et al. [100] conducted additional pre-training on ELMo [71], and BERT [22] with many different intermediate-target task combinations and tested on the GLUE. Phang et al. [72] proposed to utilize several tasks in the GLUE to fine-tune PTMs and found that MNLI [108] could improve the performance of other tasks in the GLUE. There are also some works aiming to improve the performance of specific tasks. Clark et al. [19] found that fine-tuning BERT on MultiNLI [108] can improve the performance of BoolQ. Sap et al.[82] proposed to fine-tune BERT with SocialIQA to improve the performance of CPA [79], WSC [55], and DPR [76]. SKEP [95] proposed sentiment masking and sentiment knowledge prediction objectives to fine-tune RoBERTa and

achieved better performance for many sentiment classification tasks.

As for self-supervised intermediate tasks, Gururangan et al.[33] proposed to utilize masked language model (MLM) to fine-tune PTMs on domain-specific corpora such as biomedical, computer science publication, news, and reviews, or unlabeled corpora from the datasets of target tasks.

2.3 Text Assessment

In this section, we mainly introduce two application tasks of pre-trained models on text assessment namely Automatic Short Answer Grading (ASAG) and Automated Text Scoring. Existing works about ASAG are introduced in Section 2.3.1. While various models of ATS will be shown in Section 2.3.2.

2.3.1 Automatic Short Answer Grading

ASAG is another popular task in computer-assisted assessment, which attempts to identify the correctness of the student's answer according to the correct answer as well as the given question. There is also a popular open dataset called The Joint Student Response Analysis which is the 7th task of semeval-2013 [2]. The key step of the problem is to learn more accurate semantic matching features. Existing works mainly consider two types of features namely hand-crafted features and neural features. Hand-crafted features are proposed in early works including n-gram features [38], softcardinality text overlap features [42], graph alignment features [89], averaged word vector text similarity features [89], and other shallow lexical features [67]. Recently, neural network based features are also widely utilized such as the adapted convolutional recurrent neural network (CRNN) [78], and siamese BiLSTM with earth mover's distance pooling [51]. Besides, some works also utilized combined features,

[2]https://www.cs.york.ac.uk/semeval-2013/task7.html

for example, sentence embedding, as well as token level hand-crafted features, are integrated [81].

2.3.2 Automated Text Scoring

Ke and Ng [46] summarized recent works on automated essay scoring. In general, there are three parts to solve the AES task, namely text representation learning, score mapping function, and score constraints. Almost all works utilize a linear combination function to map each text representation to a score. In the rest, we introduce various score constraints with used approaches for text representation learning.

According to different score constraints, existing works fall into three categories, namely prediction, recommendation, and reinforcement learning based models.

Prediction is the most general approach, including classification and regression. For classification, the models directly predict labels that point to different scores. In comparison, regression models constrain calculated scores to be the same as gold ones. Generally, hand-crafted features and neural network based features are two popular methods to learn text representations. Early works mainly focus on the construction of hand-crafted features such as statistical features and linguistic features. There are several early AES systems including e-rater [17], PEG (Project Essay Grade) [84], and IntelliMetric [27]. e-rater utilized ten linguistic features, including eight representing aspects of writing quality and two representing content. PEG used a larger feature set with more than 30 elements of writing quality. IntelliMetric aggregated all the features into five types, namely Focus/Coherence, Organization, Elaboration/Development, Sentence Structure, and Mechanics/Conventions. Cozma et al. [21] combined string kernel and word embeddings to extract features. With the success of deep learning, researchers start to utilize various neural networks to learn text representations. Taghipour et al. [92] explored several neural networks, such as Long Short-Term Memory (LSTM) and CNN. Finally, they found that the

ensemble model combining LSTM and CNN performs best. Dong et al. [24] proposed a hierarchical text model that utilized CNN to learn sentence representations, and LSTM was used to learn text representations. Yi et al. [93] introduced a model called SKIPFLOW, which aimed to capture neural coherence features of the text via considering the adjacent hidden states in the LSTM model.

In the recommendation view, learning to rank approaches is another popular method to solve this task. Helen et al. [114] firstly addressed this problem as a rank preference problem. Based on statistical features, RankSVM, a pairwise learning to rank model, was used as score constraint. Chen and He [14] utilized listwise learning to rank model to learn a ranking model based on several linguistic features.

Reinforcement learning based models are also possible solutions. Wang et al. [104] utilized dilated LSTM to learn text representations. Then scores calculation was guided by quadratic weighted kappa based reward function.

Chapter 3

GGP: Glossary Guided Post-processing for Word Embedding Learning

3.1 Introduction

Word embedding learning is the task to utilize a continuous vector to represent each word. With the success of Word2Vec [63] and GloVe [70], which are trained on a large corpus via a simple neural network or matrix factorization, pre-trained word representations are widely used in various Natural Language Processing (NLP) tasks, such as sequence labeling task [52], text classification [47], etc. However, typical word embedding models including Word2Vec and GloVe, are based on the Distributional Hypothesis [35], which utilizes the distribution of context words as the target word representation. In practice, the corpus is always limited, which makes it difficult to calculate the actual context word distribution of each target word. For example, synonyms and antonyms are usually hard to distinguish. Meanwhile, the quality of the word embedding highly depends on the frequency of the word in the corpus, rare

words are usually discarded.

To improve the quality of the word embedding, various knowledge bases such as WordNet [64], FrameNet [2], and Paraphrase Database [30] are considered. Meanwhile, joint optimization and post-processing are two popular approaches to incorporate domain knowledge, which have achieved better performance. Joint optimization based models design extra constraints according to domain knowledge. And they retrain a new model with integrated objectives on a large corpus and knowledge bases, which usually require much training time. As for post-processing based models, they fine-tune pre-trained word embedding models with new constraints on the knowledge bases, and the data volume of the knowledge bases is much smaller than that of the corpus. So post-processing is a more efficient way. Besides, as many pre-trained word vectors exist, e.g., Google News Vectors [1] (based on Word2Vec), GloVe [2], and FastText [3] [6], post-processing approaches are more effective.

Recent works focus on fine-tuning pre-trained word embedding models with word-to-word semantic knowledge such as synonyms and antonyms. Retrofitting [28] utilized semantic lexicons' relational information and assumed linked words should learn similar representations. ER-CNT [32] used a deep neural network to fine-tune word vectors by adding constraints on synonyms and antonyms. It shows a significant improvement in topical similarity datasets, while loses the functional information [56]. There are also some works utilize the glossary to learn word representations via extending the original corpus with word definitions. The Dict2vec [96] model constructed word pairs from both the corpus and dictionary entries. Especially for negative sampling, it ignored the words in pairs from the second part. Then Skip Gram (Word2Vec) is used to learn the word embedding model. CPAE [9] proposed a LSTM based auto-encoding model to learn word representations from dictionary definitions which are assumed to be similar to their embedding. However, Dict2vec

[1]https://code.google.com/archive/p/word2vec/

[2]https://nlp.stanford.edu/projects/glove/

[3]https://fasttext.cc/docs/en/english-vectors.html

required to retrain the word embedding model, which was time-consuming. CPAE combined multiple senses of each word into one sense, which is not reasonable. And in CPAE, the authors also introduced a post-processing model, however, it could only fine-tune the words having a definition or that occur in definitions.

Glossary is the collection of glosses, and each gloss consists of a word as well as its definition (a word sequence to explain the word). In common sense, looking up the glossary (dictionary) is one important way for humans to learn a new word, and learning from example sentences of the word is another way. However, corpus is always not enough to estimate the real context words distribution, and there are also no effective approaches for semantic composition. Combining the corpus and the glossary is necessary, and they should be modeled in different ways. In Distributional Hypothesis based word embedding models, word vector represents the distribution of context words, which combines context words of different senses. Besides learned word representations are the composition of different senses. So word embedding should be approximate to the linear combination of its senses' representations.

We propose GGP (**G**lossary **G**uided **P**ost-processing) model, which combines a general function to fine-tune all pre-trained word representations, and an auto-encoding model to learn the representation of each sense. By joint optimization, embedding learned by GGP exhibits better topical and functional similarity. In addition, to speed up the convergence of the auto-encoding model, word definitions are also used to pre-train it.

To evaluate whether our model learns both topical and functional information, we test our model on six word tpical/functional similarity datasets, i.e., Men, Simverb3500, RW, Simlex999, WS353 and Mturk. Experimental results show that our model outperforms rivals by at least 4.1% in all benchmarks. And compared to GloVe, our model outperforms more than 7%. We also use two glossaries and two pre-trained word embedding models with different vocabulary sizes to show the effectiveness of our model. In summary, our contribution are:

- For multi-sense word, we learn sense representation respectively and utilize attention to integrate them.
- We combine a general post-processing function and sense representation learning model so that each pre-trained word representation could be post-processed.
- Experimental results show that our model learn both topic and functional information and performs much better than previous model.

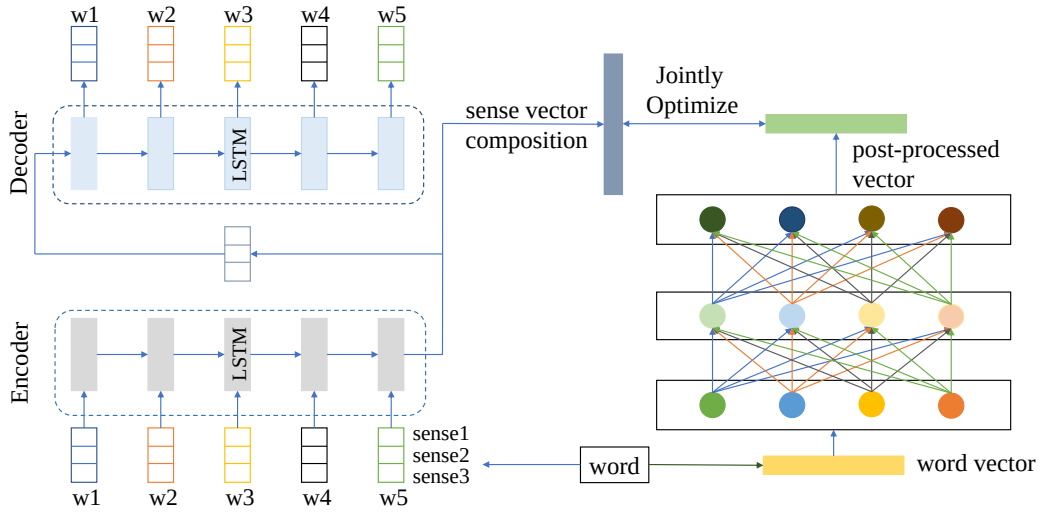


Figure 3.1: Model Framework (an example of one word with three senses)

3.2 GGP Model

Our model contains two parts as shown in Figure 6.1. The left part is a sequence to sequence auto-encoding model [57], which is used to transform each sense entry into a sense vector. The right part is a general mapping function, a multi-layer fully-connected feed-forward network, to fine-tune each word vector. Besides, we expect the general function could preserve learned information so that the word embedding will not change too much. Furthermore, We constrain the linear composition of sense vectors from the left part to be similar to the post-processed vector in the right part.

3.2.1 Sequence to Sequence Auto-encoding

The auto-encoding model consists of an encoder which transforms a sequence into a vector, and a decoder which decodes the vector to a new sequence. The encoder has two layers, the first layer is a bi-directional Long Short-Term Memory (LSTM) neural network which captures the full context information of each word. The second layer, a vanilla LSTM network, is used to learn a composition function to transform a whole sentence into one vector by utilizing word order information. The decoder is also a two-layer vanilla LSTM neural network which tries to reconstruct the input sequence from the output of the encoder.

Given a word w_t, the definition sequence of i_{th} sense is $in_t^i = \{in_{t,1}^i, in_{t,2}^i, ..., in_{t,n_i}^i\}$, where n_i is the number of words of the sense definition. $Encoder(\cdot)$ and $Decoder(\cdot)$ are the encoder model and the decoder model respectively. The encoder transforms the definition sequence into a sequence of hidden states $he_t^i = \{he_{t,0}^i, he_{t,1}^i, ..., he_{t,n_i}^i\}$ as shown in Formula 3.1. Meanwhile, he_{t,n_i}^i is used to represent the i_{th} sense. And the decoder utilizes the sense representation he_{t,n_i}^i as the initial hidden state $hd_{t,0}^i$, then generates a new sequence $hd_t^i = \{hd_{t,1}^i, hd_{t,2}^i, ..., hd_{t,n_i}^i\}$ as shown in Formula 3.2.

$$he = \text{Encoder}(in_t^i) = \text{BiLSTM}(\text{LSTM}(in_t^i)) \tag{3.1}$$

$$hd = \text{Decoder}(he_{t,n_i}^i) = \text{LSTM}(\text{LSTM}(he_{t,n_i}^i)) \tag{3.2}$$

The reconstruction loss L_r of the auto-encoding part is defined by CrossEntropy loss shown in Formula 3.3, where n_t is the number of senses, in_t^i is the input sequence of the i_{th} sense, and hd_t^i is the output sequence of the decoder.

$$L_r = \frac{1}{n_t} \sum_{i=1}^{n_t} \text{CrossEntropy}(hd_t^i, in_t^i) \tag{3.3}$$

3.2.2 Composition of Sense representations

With all calculated sense representations $he_t = \{he^1_{t,n_1}, he^2_{t,n_2}, ..., he^{n_t}_{t,n_{n_t}}\}$ of the given word, additive attention [1] is used to calculate S_{w_t}, the composition representation of all sense vectors, which is shown as Formula 3.4:

$$S_{w_t} = \sum_{i=1}^{n_t} w_i * he^i_{t,n_i} \tag{3.4}$$

where he^i_{t,n_i} is the i_{th} sense representation of the word t, w_i is the weight of the representation of i_{th} sense calculated as Formula 3.5. v_{w_t} is the word embedding of word t, besides, W and b are parameters to be learned.

$$w_i = \frac{\tanh(\boldsymbol{W}[v_{w_t}, he^i_{t,n_i}] + \boldsymbol{b})}{\sum_{j=1}^{n_t} \tanh(\boldsymbol{W}[v_{w_t}, he^j_{t,n_j}] + \boldsymbol{b})} \tag{3.5}$$

3.2.3 Multi-layer Fully-connected Feed-Forward Network

To learn a general post-processing mapping function to fine-tune any pre-trained word vectors, a multi-layer fully-connected feed-forward neural network is adopted. It maps a vector $v_{w_t} \in \mathbb{R}^{d*1}$ to a new one $v'_{w_t} \in \mathbb{R}^{d*1}$, where d is the dimension size of the word embedding. Each layer is calculated as:

$$h_n = \tanh(W_{n-1}h_{n-1} + b_{n-1}) \tag{3.6}$$

Where $\boldsymbol{W}$ and $\boldsymbol{b}$ are parameters to be learned. And the input h_0 is the pre-trained word vector v_{w_t}, the post-processed vector v'_{w_t} is h_n. To preserve the information learned by the pre-trained model, we use F_2 norm as the loss L_n shown in Formula 3.7.

$$L_n = \|v'_{w_t} - v_{w_t}\|^2_F \tag{3.7}$$

3.2.4 Extra Constraint and Joint Objective

Given post-processed word vector v'_{w_t} and sense composition representation S_{w_t}, we assume the word representation learned from the corpus and that from the glossary

should be similar. The loss of the extra constraint L_s is defined as Formula 3.8, where D means distance measurement.

$$L_s = D(v'_{w_t}, S_{w_t}) \tag{3.8}$$

All these three objectives are optimized jointly, so the total loss L is defined as Formula 3.9, where α and β are adjustable weights.

$$L = L_s + \alpha L_r + \beta L_n \tag{3.9}$$

3.3 Experiment and Discussion

3.3.1 Pre-trained Vectors

GloVe has several pre-trained word vectors with different sizes of corpora, and it outperforms other word vectors in most word similarity tasks. We select two pre-trained word vectors trained on two extremely large corpora. One is trained on the corpus of 840 billion tokens and the vocabulary has 2.2 million unique words. The other one uses the corpus containing 42 billion tokens and the size of the vocabulary is 1.9 million. All of the word vectors are in 300 dimensions.

3.3.2 Definition Entries

In our experiment, we construct two dictionaries from WordNet and Opted. We only extract the definitions explaining the words in the two vocabularies. For the dictionary from WordNet [64], we extract the word definitions with multi-senses via the interface provided by NLTK [4]. And Opted is from an open project called The Online Plain Text English Dictionary [34], we parse all the original dictionary files

[4]http://www.nltk.org/howto/wordnet.html

Vocab	Dict	**NW**	**NSW**	**ALS**
1.9M	WordNet	51621	1.57	5.39
1.9M	Opted	64616	3.49	6.10
2.2M	WordNet	47337	1.62	5.33
2.2M	Opted	58575	3.68	6.04

Table 3.1: Dictionary Statistics

downloaded from the website [5] and construct a unified dictionary, while each definition is separated into multiple senses. We also calculate some statistical properties, including the number of words having senses (NW), the average number of senses for each word (NSW), and the average length for each sense (ALS), shown in Table 3.1. We find that Opted contains more words in the two vocabularies of the pre-trained models, and provides more words to explain senses.

3.3.3 Pre-train Auto-encoding Model

In previous work [41], researchers proposed a pre-trained language model to improve classification tasks. Motivated by it, we also pre-train the auto-encoding part, then jointly optimize the auto-encoding part as well as the post-processing part. Furthermore, dictionary entries are used to train the auto-encoding model until the loss on the validation set increases.

3.3.4 Model Parameter Specification

In all experiments, we adopted softmax as a special distance measurement, which is approximated by negative sampling [63]. We separate 20% of word and definition pairs as validation samples. And we use two different word embedding matrices for

[5] http://www.mso.anu.edu.au/~ralph/OPTED/

Table 3.2: Word Similarity Experiment Results (Spearman's correlation coefficient ρ * 100)

Model	Corpus	Dict	**SV**	**SL**	**Men**	**RW**	**Mturk**	**WS353**	**WS-S**	**WS-R**
GloVe	42B	-	22.6	37.4	74.3	37.4	64.5	63.2	69.8	57.1
ER-CNT	42B	-	**47.0**	**61.1**	64.7	36.3	56.6	59.5	66.0	49.0
CPAE	42B	-	27.5	33.2	52.2	23.0	33.7	40.5	49.5	32.8
GGP	42B	WN	28.6	40.6	80.2	42.5	69.8	73.8	76.2	70.2
GGP(PT)	42B	WN	29.3	42.4	80.7	42.8	**70.3**	75.5	78.0	72.7
GGP	42B	Opted	**30.6**	43.8	80.8	43.8	68.2	**75.6**	**78.4**	72.4
GGP(PT)	42B	Opted	30.5	**44.2**	**81.2**	**44.6**	68.7	75.0	77.8	**72.7**
Glove	840B	-	28.3	40.8	80.5	45.5	69.3	71.2	80.2	64.4
ER-CNT	840B	-	**47.2**	**59.0**	67.6	43.2	62.3	59.3	70.6	44.6
CPAE	840B	-	33.8	39.7	62.3	32.4	41.7	48.7	60.3	39.3
GGP	840B	WN	32.0	42.7	**82.5**	49.0	71.5	73.8	79.0	67.9
GGP(PT)	840B	WN	31.9	44.2	82.4	49.5	**72.3**	**75.5**	**80.2**	68.2
GGP	840B	Opted	32.7	**44.3**	82.2	48.6	69.5	73.0	79.4	65.4
GGP(PT)	840B	Opted	**32.9**	44.2	82.1	**50.4**	70.1	74.7	79.9	**68.2**

auto-encoding and fully-connected networks respectively, where both the embedding size are set to 300. In fully-connected networks, we set 3 hidden layers. α and β are both set to 0.3 for two pre-trained word embedding models. For each word and definition pair, the number of negative samples is 10. And an Adam optimizer is used with an initial learning rate as 0.5. Besides, the loss of validation samples is monitored to know when to stop early.

3.3.5 Word Similarity

We tested our model on six word similarity datasets, including Men [10], Simverb3500 (SV) [31], RW [61], Simlex999 (SL) [39], WS353 [29] and Mturk [75]. Simberb3500 and

Simlex999 are topical similarity datasets [32]. Men, RW, and Mturk are functional similarity datasets. WS353 consists of a topical similarity part WS-S and a functional similarity part WS-R. Spearman's ρ rank correlation between the ground truth and calculated word similarity from word embedding is used to evaluate the performance. We show the experimental results on five models including GloVe, ER-CNT [32], CPAE [9], and our model (GGP), as well as our model with the pre-trained auto-encoding model (GGP+PT). The former three models are used as baseline models. For ER-CNT model, the paper only reported the performance on SV and SL datasets, we run the source code published by the authors [6] to obtain the performance on all six datasets. CPAE only utilized Word2Vec to evaluate their model and ignored GloVe, since they said that GloVe performs much better than Word2Vec in their paper. We also run the source code published by the authors [7] to test the model on the six datasets. Finally, we tested two pre-trained word embeddings with different vocabulary sizes and two different dictionaries.

3.3.6 Analysis and Discussion

According to Table 3.2, ER-CNT [32] outperforms other models by a large margin in SV and SL (two topical similarity tasks), however, on the rest functional similarity datasets ER-CNT performs worse than GloVe. CPAE [9] shows a little improvement in SV, and performs worse on all other datasets. Our model enhances GloVe by around 7% in all datasets, which shows that our model could capture topical and functional information simultaneously.

The Influence of Distance Measurement Absolute distance and relative distance are two common distance measurements. Absolute distance including cosine similarity, Euclidean distance, etc., stresses that two words should be similar enough. Relative distance, asks two words should be closer than another two words, for exam-

[6] https://github.com/codogogo/explirefit

[7] https://github.com/tombosc/cpae

ple, softmax constrains that the distance between the target word and context word should be smaller than that between the target word and negative words. In ER-CNT [32], "Contrasting Objective" is also a relative distance and shows significant improvement. In our model, softmax also shows a surprisingly better performance. The reason is that the relative distance tends to give a partial order to several words (more than two words), while the absolute distance only constrains two words.

The Influence of the Size of the Corpus GloVe learned from the larger corpus performs better. When GloVe trained on the small corpus was fine-tuned with the glossary, it shows a comparable, sometimes even better performance, which shows the effectiveness of utilizing the glossary. Experiments show less improvement when incorporating the glossary with the larger corpus, for the overlapping information between the corpus and the glossary increases. Opted dictionary show better performance than WordNet on the small corpus. Since Opted contains more explanations which provide more extra information.

The Influence of Pre-Trianed AE In most datasets, pre-trained auto-encoding model shows improvement. Maybe the pre-trained auto-encoding model helps the total model to be trained from a better initialization.

3.4 Summary

In this paper, we propose a model to incorporate dictionary entries to post-process word embedding to be more topical and functional. Experimental results show the effectiveness of our model. To further improve the work, we will combine various word relationships in a partial order to improve the word embedding models.

Chapter 4

Automated Post Scoring: Measuring Post-Topic Relevance and Post's Writing Quality in Online Forum Discussion for Learning Performance Estimation

4.1 Introduction

Online education has shown significant growth over the last decade especially under the pandemic of COVID-19 [69]. As one of the most important features of online education, discussion forum brings various benefits including boosting learning performance, reducing dropout rates, and increasing course satisfactory [116]. So many instructors adopt online discussion to assess students' writing mechanics and knowledge understanding from discussion posts [109]. However, marking numerous posts is time-consuming and labor-intensive for instructors [92]. Mutual disagreement often

occurs when multiple evaluators marking same posts. Even for a single evaluator, grading consistency is hard to guarantee given numerous posts [85]. Therefore, there is an urgent need for automated post scoring (APS) to evaluate writing mechanics and knowledge understanding of students by grading their posts according to given topics and quoted posts automatically. An illustration of the task is shown in Figure 4.1. It is difficult to assess the knowledge understanding of students directly, so we evaluate the relevance between posts and given topics instead. In addition, quoted posts are used as auxiliary features to enhance posts evaluation.

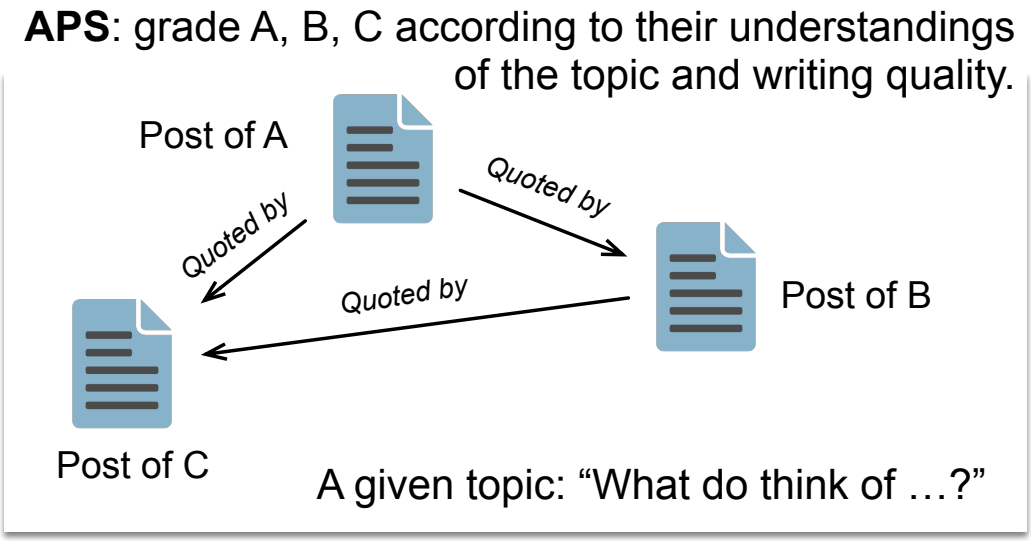


Figure 4.1: Illustration of APS

Previous two types of works focused on evaluating either the writing quality or the correctness of short answers. Automated Text/Essay Scoring (AES) [92] mainly evaluates the writing quality of the independent long essays or texts. We refer to ideas in these works to measure the writing quality of posts. Some works also measure the relevance between text and prompts. These models utilize simple statistic features which are difficult to capture deeply semantic matching. They also face difficulties to align the semantics of multiple sentences. While Automatic Short Answer Grading Task (ASAG) [65] measures the correctness of the short student's answer according

to the correct answer and given question. The semantic matching between long texts (hundreds of words) is more complicated than that of short texts (one or two sentences). Unlike prompt-relevant AES/ATS, APS is required to utilize extra quoted posts to enhance the measurement of the posts. And inspired by new neural semantic matching methods [13], advanced semantic matching approaches are necessary. Compared with ASAG, APS pays attention to the writing quality of long texts, the relevance between long texts, and the quoted posts. Examples from three typical dataset corresponding to three tasks are shown in Table 4.1. And the differences between three tasks are summarized in Table 4.2.

In this paper, we propose different methods to incorporate topics and quoted posts respectively, since they play different roles in discussion forums. During the discussion, students write posts to respond to the topics, so the relatedness between posts and topics illustrates their knowledge understanding. It is possible to use semantic matching to measure relatedness. While students quoted partial arguments of other students to support or explain their arguments. It is necessary to extract supporting arguments from quoted posts as auxiliary features.

Our vision, however, entails three challenges when applied to reality. The first challenge is how to augment topics. They are too short and abstract so that it is difficult to directly measure the relatedness between abstract concepts and detailed arguments. The second challenge is how to measure the relatedness between long posts and long topics. Posts usually introduce several arguments, and the order of the arguments may be different from that of the concepts in the topics. The last challenge is how to extract supporting arguments. Quoted posts contain many arguments, however, not each argument is useful to support the students' arguments.

To tackle these challenges, we use data augmentation methods to extend the topics, and propose two different models to integrate topics and quoted posts. More specifically, we extend the topics with text contents obtained from the hyperlinks contained in given topics' description. To map the arguments in posts and concepts description

Table 4.1: Sample data of ODD, ASAP Dataset, and Semeval 2013 task 7. For page limit, only part of posts are shown in Post and Quoted Post in ODD.

APS	**AES/ATS**	**ASAG**
Topic: "If we could record the activity of all neurons, we could understand the brain." Gero Miesenboeck (2010) @ TED. Based on the arguments presented by Gero Miesenboeck (TED Video: Hyperlink 1) Partha Mitra (Scientific American Letter: Hyperlink 2), would you agree with the above statement? What are your rationale(s), with reference to your textbook the psychology literature, that support your stand?" **Post**: Thanks *2015_1_S202* for taking the role to summarize our points. It's nice to see your response. I wonder which of the stance you stand for. As you point out that our discussion is mainly focusing on the feasibility of the recording method, I would like to add something to support my argument. In order to crack the neural code, understanding how single neurons and complex networks process perceptions is a vital factor to understand the brain. **Quoted Post**: What i can conclude from your discussions is that *2015_1_S57* argued that the patterns of our brains are complicated and everchanging and the insufficiency of recording the activity of our brains obstructs the understanding of our brains. *2015_1_S72* pointed that measuring the whole brain and then decoding then is not feasible. *2015_1_S292* thought that alternative ways to understand the brains through psychology. **Score**: 15.5	**Post**: the essay rough road ahead: do not exceed posted speed limit describes a mans bicycle ride through california. now, california is very hot during the summer, which is when the cyclist is riding. this setting greatly affects the mans journey. it made it very difficult for him to finish his ride. he drank most of his water in the beginning of his ride so he gets very dehydrated. the text states, the water bottles contained only a few tantalizing sips. as you can see the setting makes this mans bikeride very hard. **Score**: 2.0	**Quetion**: Explain why you got a voltage reading of 1.5 for terminal 1 and the positive terminal. **Correct Answer**: Terminal 1 and the positive terminal are separated by the gap. **Student Answer**: because terminal one and the positive terminal are connected **Label**: correct, *contradictory*, incorrect

Table 4.2: The difference between APS, AES/ATS and ASAG.

Task	Typical Dataset	Input Data	Text Length	Labels
APS	ODD	posts, topic, quoted posts	hundreds of words	real number
AES/ATS	ASAP [1]	text	hundreds of words	real number
ASAG	Semeval-2013 [2]	question, correct answer, student answer	one or two sentences	several classes

in topics, we propose a matching model which learns sentences' representations firstly and calculates the interactions between any two sentences to extract the matching features. To extract the supporting arguments from quoted posts, we propose a representation model which uses an attention model to calculate the weighted sum of the sentences' representations of quoted posts. In addition, we also adopt hierarchical text models to learn the syntactic and semantic information of the posts. We combine these three models as a mixture model and extract features to predict the posts' scores. We conduct various experiments to verify the effectiveness of topic augmentation, and incorporating topics and quoted posts. Our mixture model outperforms the hierarchical text model that only assesses the posts by a large margin, nearly 9 percent in Quadratic Weight Kappa.

Our contribution could be summarized as follows:

1. We propose a new task called APS, which evaluates the writing quality and relevance of posts with extra topics and quoted posts.

2. To solve the task, we propose to measure the relevance by the semantic consistency between the posts and the topics and enhance posts prediction with the related supporting arguments extracted from quoted posts.

3. Experimental results show that the measurement of relevance and writing qual-

ity can score the posts much more accurately, nearly 9 percent in QWK.

4.2 Problem Definition and Dataset

In this section, we will introduce some basic concepts in the online forum. Then we give a formal problem definition and show more details about our Online Discussion Dataset.

4.2.1 Glossary of Online Forum

In this section, we will explain some basic terminologies that are widely used in online forums, including post, thread, and quoted post.

- Post: A post is a user-submitted text enclosed into a block containing the user's details and the date and time it was submitted, which aims to respond to the given topic.
- Thread: A thread is a collection of posts that respond to the same topic, usually displayed from oldest to latest.
- Quoted Post: Quoted post is the post that was quoted by another post in the same thread.

4.2.2 Problem Definition

In this section, we will define the problem from each student's perspective. In the online forum, there are several threads. And for each thread, a topic $T(s_i)$ is given firstly by the instructors. Then for each students s_i, he/she is asked to submit n_i posts $P(s_i) = \{P(s_i)_1, ..., P(s_i)_{n_i}; n_i \geq 1\}$ according to the given topic. During the discussion, the student usually quotes other students' arguments such as "Refer to

Table 4.3: Statistics of Online Discussion Dataset

#Students	#Posts	AP	APL	ATL	#Quoted	AQ	Range
694	2542	3.66	270	41.19	405	1.21	[5.0,20.0]

XXX's point", so all the posts of quoted students are used as quoted posts. For each post $P(s_i)_j$, it quotes several students' posts $q(i,j) = \{P(s_k) : s_k \in S(i,j)\}$, where $S(i,j)$ is the students set quoted by the post. All quoted posts are $Q(s_i) = \{q(i,1), ..., q(i,n_i)\}$. Instructor marks all posts of each student with a numerical value $G(s_i)$. So the proposed task is to learn a mapping function mapping $T(s_i)$, $P(s_i)$, and $Q(s_i)$ to $G(s_i)$ as shown in Formula 4.1.

$$\min(G(s_i) - f(T(s_i), P(s_i), Q(s_i)))^2 \tag{4.1}$$

4.2.3 Dataset Construction and Pre-processing

We cooperate with the Department of Applied Social Sciences in our university and get all the online discussion posts from the course "Introduction to Psychology" in one academic year (two semesters). Based on these posts, we construct an Online Discussion Dataset, which contains 86 sub-forums (threads). In each sub-forum, the instructor gave a topic description first, which is a social science problem with several keywords and many hyperlinks. With the given topic, students were asked to write their arguments about the topic with references, which are similar to academic writing. During writing the posts, students were also encouraged to quote other students' arguments to explain their points. Some statistics are given in Table 4.3. In the table, *#Students* means the total number of students in our dataset, *#Posts* means the total number of posts, and AP means the average number of posts for each student. Meanwhile, APL is the average number of words of each post, ATL is the average number of words of each topic, *#Quoted* is the total number of students who

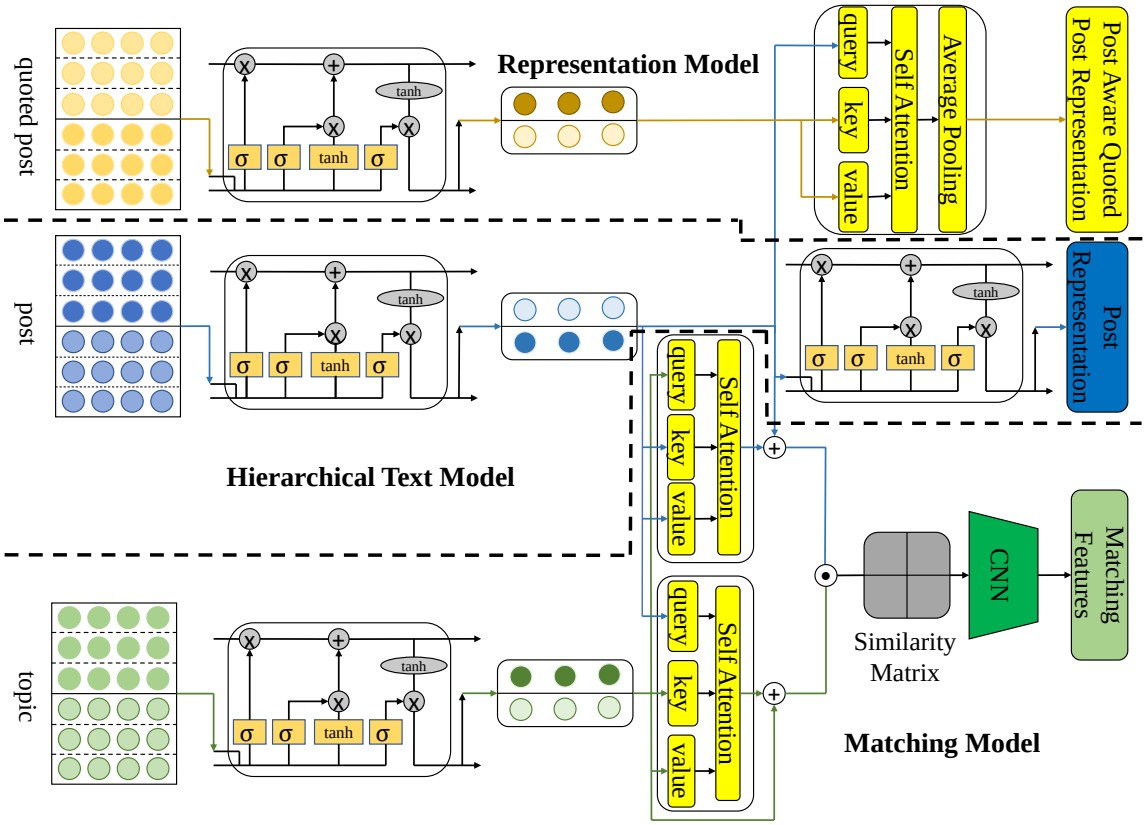


Figure 4.2: Framework of our mixture model

quoted other students' views, and AQ means the average number of quoted students for each student. Finally, *Range* is the score range.

With the raw posts data, our pre-processing mainly includes two parts. Firstly, to preserve privacy, we need to mask all the student names that appeared in the posts, and map them to the real students in the sub-forum. To identify the students' names, and ignore the names in the references, we use a regular expression to replace all the references into special tokens, and a tool of Named Entity Recognition is used to find names in text. Besides, fuzzy string matching is used to map identified names to real students in the forum. To make sure the complete masking, human verification is also necessary. Then, there are lots of hyperlinks and references in each post to support students' arguments, we replace various hyperlinks and references with special tokens, similar to replace the names and numbers with special tokens in AES tasks. In the dataset, each student submits several posts to show his/her arguments and quoted several students' posts. We combine all the posts of each student into one, so as the quoted posts.

Data augmentation is also adopted, since the topic description only consists of several keywords and hyperlinks. And the abstract concepts are hard to understand by the machine. We try to enrich the topic with the text provided by the hyperlinks such as the text content of the web pages or the subtitles of the videos. For those topics without hyperlinks, we search the abstract concepts in Wikipedia [2] and add the text content of the web pages into the topics' description. With the aforementioned processing, the average topic length is extended from 41.19 to 576.37. Enriching the topic description also improves the performance of post scoring as shown in Section 5.3.4. In Table 4.4, we show some examples of topic extension.

[2]https://en.wikipedia.org/wiki/Main_Page

Table 4.4: Examples of topic extension via recording subtitles of the video or searching the keywords in wikipedia

Original Topic	Augmented Topic
Based on this discussion on MBTI types (**Hyperlink**), to what extent is preference towards Anime / Comic / Game (ACG) being influenced by one's personality?	**Content of Hyperlink**: According to an analytical psychologist, Carl Gustav Jung's theory of psychological types, there are 4 perceiving psychological functions: A. Introverted sensing This function is affected by hormone levels inside one's body, and one very often seeks sense of security there. Such a function forms Keirseyian temperament called Guardian, and this is commonplace in all ordinary ACG fans. The relevant MBTI types for this temper are ISFJ, ISTJ, ESFJ, ESTJ. Some may call them Pure MK because they seek order on-duty and break order off-duty. B. Extraverted Sensing This function allows for originality in production, and this facilitates conceptualization of ideas as well.
Evidence from psychology literature suggests that **human memory** is far from being very accurate and we are prone to process memory with reference to our biases. From this perspective, could we argue that human memory should be considered inferior to memory in computers and other machines?	**Content from Wikipedia**: Memory is the faculty of the brain by which data or information is encoded, stored, and retrieved when needed. It is the retention of information over time for the purpose of influencing future action. If past events could not be remembered, it would be impossible for language, relationships, or personal identity to develop. Memory loss is usually described as forgetfulness or amnesia. Memory is often understood as an informational processing system with explicit and implicit functioning that is made up of a sensory processor, short-term (or working) memory, and long-term memory. This can be related to the neuron. ...

4.3 Posts Assessment Model

In this section, the whole framework is illustrated first. Then, we introduce the hierarchical text model first. Also, we show the cross attention model which is used in the latter two models. Finally, we introduce the matching model and the representation model respectively.

The whole model framework is shown in Figure 6.1. The hierarchical text model is in the middle of the picture which provides the representations of sentences from posts to the representation model and matching model. The bottom part is the matching model capturing the relevance of the topic and post at the sentence level. And a representation model integrating quoted posts is on the top. All input examples are two sentences with 3 words in each, LSTM is used as an example to learn sentence representations and post representations. Cross attention is used to calculate the similarity matrix in the bottom part, as well as select sentences from quoted posts in the top part. Finally, the matching feature, post representation, and post aware quoted post representation are concatenated together to calculate the student score.

4.3.1 Hierarchical Text Model

In general, there are two ways to represent a long text, the first one treats the long text as a long word sequence, and the latter one constructs a hierarchical model which regards the long text as a sentence sequence, and for each sentence, it is also a word sequence. In our scenario, students usually write the first post to introduce their main argument, and the rest post to explain and add extra references. So we combine all the posts from the same student into one, and each combined post may have nearly 1000 words on average. In general, there are three sequence models including Long Short-Term Memory (LSTM), Convolutional Neural Network (CNN), and Transformer. LSTM always faces some difficulties when coping with very long sequences for limited

memorization ability. And coping with long sequence, efficiency is always the most significant problem for transformer, such as BERT, a transformer-based pre-train language model, sets the maximum sequence length to 512. And CNN model will lose word order information, which is essential to learn text semantic representation. Since the text is too long, it is unreasonable to adopt the first representation method. In this paper, we adopt the second method, the hierarchical text model to learn the post representation.

In the hierarchical model, a long text is tokenized into several sentences, and each sentence is tokenized into several words. For efficiency reasons, we use LSTM or CNN as the semantic composition model to learn the sentence representations from the word representations. Besides, similar semantic composition models are used to learn the text representations from the sentence representations. In the rest, we will mainly introduce how to learn the sentence representation, since the method to learn the text representation is the same.

To learn the semantic representation of each sentence $p_i = \{w_{i,1}, ..., w_{i,m_i}\}$, where m_i is the number of words, a word embedding matrix E is constructed first. Then each word $w_{i,j}$ is mapped into a vector $E(w_{i,j})$ via the embedding matrix. With obtained word vectors, LSTM or CNN is used to combine word semantics into the sentence representation.

LSTM could capture the word order information to learn the syntactic information. Meanwhile, compared with long text, each sentence only has a pretty small number of words, and the gate mechanism in LSTM is powerful enough to cope with these sequences with limited length. To calculate the sentence representation $R_s(p_i)$ via LSTM, we obtain all the hidden states $H(p_i) = [h_0, h_1, ..., h_{m_i}]$ as shown in Formula 4.2, then average pooling or attention mechanism is used to combine these hidden states.

$$H(p_i) = \text{LSTM}(\{E(w_{i,1}), E(w_{i,2}), ..., E(w_{i,m_i})\}) \tag{4.2}$$

The advantage of average pooling is to capture more early information in each sequence, which will gradually decrease in RNN based models. However, not each word contributes equally to the sentence semantic. Unlike average pooling, attention is proposed to learn different weights for each word. The calculation of the attention weight is shown in Formula 4.3, where $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ are projection matrices. And the sentence representation could be computed by Formula 4.5.

$$\boldsymbol{Wt} = \text{softmax}(\boldsymbol{W_2} \cdot \tanh(\boldsymbol{W_1} \cdot H(p_i)^T)) \tag{4.3}$$

$$\text{softmax}(\overrightarrow{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{4.4}$$

$$R_s(p_i) = \boldsymbol{Wt} \cdot H(p_i)^T \tag{4.5}$$

CNN can capture phrase information which is useful to identify abstract concepts in posts. With obtained word embedding sequence $E(p_i)$, convolutional layer is computed as Formula 4.6, where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the parameters and shared across all windows in the sequence. Relu is used as the non-linear activation function and maximum pooling is added to compress the learned features as shown in Formula 4.7.

$$\text{Conv}(E(p_i)) = \boldsymbol{W} \cdot E(p_i) + \boldsymbol{b} \tag{4.6}$$

$$R_s(p_i) = \text{MaxPooling}(\text{Relu}(\text{Conv}(E(p_i)))) \tag{4.7}$$

With learned sentence representations of each post $R_s(p) = \{R_s(p_1), ..., R_s(p_m)\}$, where m is the number of sentences, Formula 4.5 and Formula 4.7 are also used to learn the post representation $R(p)$ by replacing the words embedding sequence $E(p_i)$ with the sentences representations sequence $R_s(p)$ as shown in Formula 4.8 and Formula 4.9. Attention mechanism and LSTM could be used together to learn the importance of different sentences.

$$R(p) = \text{AvgPooling}(H(R_s(p))^T) \tag{4.8}$$

$$R(p) = \text{MaxPooling}(\text{Relu}(\text{Conv}(R_s(p_i)))) \tag{4.9}$$

In this section, we introduced the whole process that how to implement the hierarchical text model by various combinations of word representations composition and sentence representations composition.

4.3.2 Cross Attention Model

Referring to previous work [118], cross attention is a key approach to model the word-level interaction between two sentences or sentence-level interaction between two texts. Cross attention is a variation version of self-attention proposed in Transformer [98].

Cross attention also has three input sequences, namely $Q = [R_s(p_i)]_{i=0}^{n_Q-1}$, $K = [R_s(p_j)]_{j=0}^{n_K-1}$, $V = [R_s(p_k)]_{k=0}^{n_V-1}$ respectively, where n_Q, n_K and n_V denote the number of sentences in each long text, and $R_s(\cdot)$ stands for the sentence representation model, n_K is equal to n_V. The model first takes each sentence in the query text to attend to sentences in the key text via Scaled Dot-Product Attention [98]. Then those attention results were applied upon the value text, which is defined as:

$$\begin{aligned} \text{Att}(Q, K) &= [\text{softmax}(\frac{Q[i] \cdot K^T}{\sqrt{d}})]_{i=0}^{n_Q-1} \\ V_{att}(Q, K, V) &= \text{Att}(Q, K) \cdot V \in R^{n_Q \times d} \end{aligned} \tag{4.10}$$

where $Q[i]$ is the i_{th} sentence representation in the query text Q and d is the representation size. Each row of V_{att}, denoted as $V_{att}[i]$, stores the fused semantic information of sentences in the value text that possibly have dependencies to the i_{th} sentence in query text. For each i, $V_{att}[i]$ and $Q[i]$ are then added up together, compositing them into a new representation that contains their joint meanings.

With the detailed implementation of cross attention, the matching model and representation model will be introduced respectively.

4.3.3 Matching Model

Instructors evaluate students' posts by estimating the relevance between the post and the topic. In this paper, the matching model attempts to calculate the semantic matching features to capture the semantic consistency between the post and the given topic. More specifically, with learned sentences' representations, a cross attention model calculates the interactions between any two sentences to obtain the similarity matrix. Then, CNN is used to extract the matching features.

We utilize cross attention to calculate a similarity matrix $Sim_{p,t} \in \mathbb{R}^{n_p \times n_t}$ between post sentences $R_s(p) = R_s(P(s_i))$ and topic sentences $R_s(t) = R_s(T(s_i))$ via Formula 4.11. Similar to Formula 4.7, a two dimensional convolutional layer and a maximum pooling layer are used to extract the matching feature as shown in Formula 4.12.

$$\begin{aligned} a &= V_{att}(R_s(p), R_s(t), R_s(t)) \\ b &= V_{att}(R_s(t), R_s(p), R_s(p)) \\ a &= R_s(p) + a \\ b &= R_s(t) + b \\ Sim_{p,t} &= a \cdot b^T \end{aligned} \tag{4.11}$$

$$\begin{aligned} &\text{Conv2D}(\text{Sim}_{p,t}) = \boldsymbol{W} \cdot \text{Sim}_{p,t} + \boldsymbol{b} \\ &R(p,t) = \text{MaxPooling2D}(\text{Relu}(\text{Conv2D}(Sim_{p,t}))) \end{aligned} \tag{4.12}$$

In summary, to measure the semantic consistency between posts and given topics, the cross attention model is used to calculate the semantic matching matrix, then CNN is used to extract the matching features.

4.3.4 Representation Model

Quoted posts are also important parts to measure students' posts. Since the quoted arguments in quoted posts reveal the student's understanding of the given topics.

The key idea to incorporate quoted posts is similar to the attention mechanism, we select the most relevant sentences from quoted posts as the auxiliary post aware quoted post representation. More specifically, making quoted posts and posts attend to each other, it is significant to capture dependencies between those latently matched segment pairs, which can provide complementary information for post representation. In addition, the cross attention model will not consider the sentence order, we combine all the quoted posts into one post.

With calculated sentence representation sequence for the post $R_s(p) = R_s(P(s_i))$ and that for the combined quoted post $R_s(q) = R_s(Q(s_i))$, quoted post aware post representation $R(p|q)$ is calculated by the cross attention model as shown in Formula 4.13.

$$R(p|q) = \text{AvgPooling}(V_{att}(R_s(p), R_s(q), R_s(q))) \tag{4.13}$$

In this section, the cross attention model is used to select sentences from quoted posts referring to posts.

4.3.5 Scoring Function

With learned post representation $R(p)$, matching feature $R(p,t)$, and post aware quoted post representation $R(p|q)$, we get the final representation $R_f(p) = [R(p), R(p,t), R(p|q)]$ via combining the post representation and additional features.

Then a fully connected neural network FCNN$(\cdot)$ is used as the score mapping function. In addition, $\sigma = \text{Sigmoid}(\cdot)$ activation function is used to normalize the score into [0,1] as shown in Formula 4.14. More specifically, FCNN is a linear function, $\boldsymbol{W}$ is the weight matrix and $\boldsymbol{b}$ is the bias. To learn better parameters, the mean score of all students in training set is used to initialize the bias $\boldsymbol{b}$.

$$G(s_i)' = \sigma(\boldsymbol{W} \cdot R_f(p) + \boldsymbol{b}) \tag{4.14}$$

For regression problem, Mean square error is used as the loss of the neural networks.

With all the gold scores $\{G(s_i)|i \in [1, N]\}$ and calculated scores $\{G(s_i)'|i \in [1, N]\}$ for each student, the objective function is Formula 4.15.

$$L = \text{MSE}(G(s_i), G(s_i)') = \frac{1}{N}\sum_{i=1}^{N}(G(s_i) - G(s_i)')^2 \tag{4.15}$$

4.4 Experiment

4.4.1 Experiment Setting

In this paper, we utilize 5-fold cross-validation to evaluate all models with a 60/20/20 split for train, validation, and test sets. All models are trained for 100 epochs and the best model based on the performance on the validation set is selected to evaluate the performance on the test set. We tokenize the text into sentences and words using NLTK [3], and normalize all scores range to [0,1]. The scores are rescaled back to the original scale for calculating scores of Quadratic Weighted Kappa, Spearman Correlation Coefficient, Pearson Correlation Coefficient, and Rooted Mean Square Error. GloVe [4] [70] is used to initialize the word embedding matrix, and the dimension is set to 300. Besides, the learning rate is set to 0.00015, and the batch size is set to 16.

4.4.2 Evaluation Metrics

We employ two types of evaluation metrics, namely correlation based measurements including Quadratic Weight Kappa (QWK), Spearman Correlation Coefficient (SCC), and Pearson Correlation Coefficient (PCC), as well as residuals based measurements, Rooted Mean Square Error (RMSE).

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

[4]https://nlp.stanford.edu/projects/glove/

QWK measures the agreement between human raters and machine raters quadratically. As the calculation shown in [97]. A weight matrix W is constructed firstly as shown in Formula 6.11, where i and j are the reference rating (the gold ones assigned by human annotator) and the hypothesis rating (calculated by the machine system) respectively, and N is the number of possible ratings.

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{4.16}$$

In addition, a matrix O is also calculated such that $O_{i,j}$ denotes the number of students who obtained a rating i by the human annotator and a rating j by our model. Another matrix E with the expected count is calculated as the outer product of histogram vectors of the two ratings. The matrix E is then normalized such that the sum of elements in E is the same as that of elements in O. Finally, with given matrices O and E, the QWK score is calculated according to Formula 6.12.

$$\kappa = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}} \tag{4.17}$$

SCC and PCC are two popular correlation measurements to compare the orders imposed by gold and system scores over all students [**?**]. More specifically, the Pearson coefficient, commonly denoted by ρ, is defined as the covariance of the two variables divided by the product of their respective standard deviations as shown in Formula 4.18. $\mathrm{cov}(X, Y)$ refers to the covariance of the two variables, and Σ means the standard deviation.

$$\rho_{X,Y} = \frac{\mathrm{cov}(X, Y)}{\Sigma_X \Sigma_Y} \tag{4.18}$$

The Spearman coefficient is calculated by applying the Pearson coefficient to rank transformed data. Both are unaffected by linear transformations of the data. Given vectors x and y, respectively sampling X and Y and each of length n, the sample Pearson coefficient $r_{x,y}$ is obtained by estimating the population covariance and standard deviations from the samples, as defined in Formula 4.19. Here $\bar{x}$ and $\bar{y}$ denote the sample means.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{4.19}$$

RMSE is the root of MSE loss we used, which evaluates the residuals between gold scores and calculated scores as shown in Formula 4.20.

$$\text{RMSE}(y, y') = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2} \tag{4.20}$$

4.4.3 Experiment Results and Analysis

In this section, we conduct three experiments, the first experiment shows the results of different text representation models that only utilizing the posts information. The second experiment shows the effectiveness of integrating given topics or quoted posts. The last experiment illustrates the results of combining given topics and quoted posts. Furthermore, we extend the topic description during pre-processing, we also verify the effectiveness of the extension. Since APS is a new application that considers given topics and quoted posts during marking students' online posts, in our experiment, we only verify the effectiveness of our model to integrate given topics and quoted posts.

In general, we have three types of models including hierarchical text models, models integrating topics or quoted posts respectively, and models incorporating topics and quoted posts simultaneously. Names of all models are in the form of **A**-**B** for hierarchical text models, and **A**-**B**-**D** for the last two types of models. **A** refers to the word composition model and the sentence composition model including **L** means **L**STM, **LL** means **L**STM for word composition and sentence composition, **LC** refers to **L**STM for word composition and **C**NN for sentence composition, **CL** means **C**NN for word composition and **L**STM for sentence composition. **B** represents the methods that combine the hidden states calculated by LSTM. **AP** means average pooling, and **Attn** means attention methods. **D** shows the representation model or/and matching model used by given topics or/and quoted posts. **T** means integrating **T**opics and **Q** means incorporating **Q**uoted posts. **R** refers to the **R**epresentation model and **M** refers to the **M**atching model. For example, **RQ** uses the **R**epresentation model to integrate **Q**uoted posts, **MT** utilizes the **M**atching model to incorporate given

Topics, **RQT** means that using the **R**epresentation model to integrate **Q**uoted posts and **T**opics, and **MTRQ** refers to incorporating given **T**opics and **Q**uoted posts by the **M**atching model and **R**epresentation model respectively.

Baseline Models

In this subsection, we introduce a basic text representation model, four hierarchical text representation models, and one pre-trained model that only using the posts. The basic text representation model treats each student's posts as a long word sequence, LSTM as well as average pooling is used to learn the text representation. The four hierarchical text models utilize CNN, LSTM to compose word sequences or sentence sequences respectively. Meanwhile, average pooling and attention are used to obtain the sentence or text representations from the hidden states, the output of the LSTM. In addition, we also adopt a widely used pre-trained model, RoBERTa [60], to learn text representations. All these models will be adopted as baseline models.

- **L-AP** LSTM is used to learn the whole text representation, to alleviate the poor memorization ability, average pooling is used on all the hidden states. More specifically, a two-layer unidirectional LSTM is adopted, and the dimension of the hidden state is set to 300.
- **LL-AP** Two two-layer unidirectional LSTM are used to learn sentence representations and post representations. Also, average pooling is utilized to combine the hidden states from the output of word sequences or that of sentence sequences. And 300 is the dimension size of hidden states.
- **LL-Attn** Unlike the previous model, the attention mechanism is used to combine all the hidden states of the word representations. We utilize the additive attention mechanism with an additional linear transformation layer following calculated attention weights, which has been proved to be useful in representation learning.

Table 4.5: Experiment results of the basic text model and four hierarchical text models.

Model	**QWK** ↑	**SCC** ↑	**PCC** ↑	**RMSE** ↓
L-AP	0.410	0.459	0.459	2.36
LL-AP	0.440	0.436	0.470	2.38
LL-Attn	0.443	0.437	0.475	2.35
CL-AP	0.474	**0.518**	0.523	**2.25**
LC-AP	0.426	0.448	0.463	2.47
RoBERTa	**0.511**	0.494	**0.542**	2.27

- **CL-AP** CNN is used to learn sentence representations, and LSTM is utilized to learn text representations via average pooling. For the CNN model, 100 filters is used and 2 is the filter size.

- **LC-AP** LSTM is used to learn sentence representations via average pooling, and CNN is utilized to learn text representations. 1-D convolutional network is used to combine adjacent sentences. We set the filter size to 2, 3, 4 and also use 100 filters.

- **RoBERTa** $\text{RoBERTa}_{\text{BASE}}$ is used to learn the semantic representation of each student's posts. Since the maximum length of the input sequence is 512 tokens, we combine all the posts of each student into one and extract the first 510 tokens. More specifically, we set the batch size to 8, and the learning rate to 3E-5. We fine-tune the model for 30 epochs.

The results of these six models are shown in Table 4.5. Firstly, the hierarchical model LL-AP performs comparatively even better than the pure text model L-AP. The results show that although the average pooling can alleviate the poor memorization ability, the hierarchical model can largely avoid the shortage of LSTM coping with long sequences. CL-AP achieves the best performance on two evaluation metrics

namely SCC and RMSE, and outperforms the other three hierarchical models by a large margin. More specifically, at least 3 percent in QWK, nearly 6 percent in SCC, and approximately 5 percent in PCC. With the analysis of the posts data, students usually utilize various terminology to explain and support his/her points, CNN performs better to learn the phrase information. Meanwhile, the posts are required to be organized logically, LSTM is good at capturing the order information. Pre-trained models have gained great success in natural language understanding and generation tasks, since they can capture deeply semantic meaning of the input sequences. With the only first 510 tokens of each student's posts, RoBERTa also achieves much higher performance on QWK and PCC, which outperforms the best hierarchical model, CL-AP, more than 3 point on QWK, and near 2 point on PCC. The results verified the significant effectiveness of the pre-trained model on representation learning.

Models using Topics and Quoted Posts respectively

In this subsection, we mainly introduce how the representation model and the matching model utilize given topics and quoted posts respectively. And conducting experiments on these models to verify the effectiveness of incorporating extra texts such as given topics or quoted posts. Also, we verify that the matching model is more suitable to use the topic information, and the representation model takes advantage of integrating quoted posts.

In these models, the hierarchical models are used to learn the representation of posts. And we only use LSTM to learn the sentence representations of given topics and quoted posts. With given hierarchical text models, we show how to incorporate given topics and quoted posts by the representation model and matching model respectively. Based on the three hierarchical models, LL-AP, CL-AP, LC-AP, there are three representation models to use given topics namely **LL-AP-RT**, **CL-AP-RT**, **LC-AP-RT**, and three representation models to utilize quoted posts including **LL-AP-RQ**,**CL-**

Table 4.6: Experiment results of matching and representation models.

Model	QWK ↑	SCC ↑	PCC ↑	RMSE ↓
RoBERTa	0.511	0.494	0.542	2.27
LL-AP	0.440	0.436	0.470	2.38
LL-AP-RT	0.444	0.482	0.504	2.34
LL-AP-MT	0.524	0.539	0.556	2.21
LL-AP-RQ	**0.538**	**0.559**	**0.572**	2.16
LL-AP-MQ	0.517	0.538	0.551	2.26
CL-AP	0.474	0.518	0.523	2.25
CL-AP-RT	0.461	0.507	0.509	2.28
CL-AP-RQ	0.501	0.541	0.559	**2.15**
LC-AP	0.426	0.448	0.463	2.47
LC-AP-RT	0.446	0.464	0.498	2.33
LC-AP-RQ	0.482	0.511	0.528	2.32

AP-RQ, **LC-AP-RQ**. Since the LL-Attn model has much more parameters than the representation or matching models, the combined models are hard to converge. We did not show the results of LL-Attn based representation or matching models.

The matching models utilize CNN to extract the matching features, if the matching models are combined with CL-AP or LC-AP, the two CNN models lead to hard convergence. So for the matching model, we only show the results based on LL-AP. For all the matching models, the 2-D convolutional network is used to extract matching features from the similarity matrix, to obtain features from different scales, we set the filter sizes to $3 * 3$, and $4 * 4$, and we use 100 filters. **LL-AP-MT** is the model to integrate given topics and **LL-AP-MQ** is the model to utilize quoted posts. Table 4.6 shows the experimental results of all mentioned models.

By utilizing topic information, all models including LL-AP-RT, LC-AP-RT, and LL-AP-MT, consistently perform better than corresponding hierarchical text models on

all evaluation metrics. Especially, LL-AP-MT outperforms all hierarchical text models on three correlation metrics by a large margin, more than 4 percent, and also gains the lowest RMSE score. All these results show that the topic is important to learn more accurate post scores. Besides, LL-AP-MT also outperforms LL-AP-RT, CL-AP-RT, and LC-AP-RT, more than 6 percent on the QWK score, 3 percent on the SCC score, and 4 percent on the PCC score, and obtains the lowest RMSE score. These results prove that the semantic matching model is more suitable for coping with the topic information compared with the representation model. Because CNN based matching model captures sentence-level semantic interactions (the similarity matrix calculated by cross attention) which are helpful to measure the relevance between posts and given topics. Compared with the pre-trained model, RoBERTa, LL-AP-MT also gains better performance on all evaluation metrics. It also proves the effectiveness of incorporating extra topics via the matching model.

To integrate quoted posts, LL-AP-RQ, CL-AP-RQ, LC-AP-RQ, and LL-AP-MQ, consistently outperform corresponding hierarchical text models on all evaluation metrics. More specifically, CL-AP-RQ correlates better more than 2 percent on the QWK score, 2 percent on the SCC score, and 3 percent on the PCC score. The rest three models correlate better more than 5 percent on the QWK score, 6 percent on the SCC score, and 6 percent on the PCC score. They also gain lower RMSE scores than that of corresponding hierarchical text models. All these results illustrate that quoted posts are also important to improve the accuracy of predicting students' scores. Furthermore, LL-AP-RQ outperforms LL-AP-MQ on all correlation evaluation metrics by a large margin, more than 2 percent. These results prove that compared with the semantic matching model, the representation model is more effective to integrate quoted posts. Since the representation model used cross attention to select relevant sentences in quoted posts which reveal the student's understanding of quoted posts. Compared with the pre-trained model, RoBERTa, LL-AP-RQ also gains much better performance on all evaluation metrics. It also proves the effectiveness of incorporating

Table 4.7: Experiment results of mixture models.

Model	**QWK** ↑	**SCC** ↑	**PCC** ↑	**RMSE** ↓
LL-AP-RQ	0.538	0.559	0.572	2.16
LL-AP-RTQ	0.551	**0.587**	0.584	2.17
CL-AP-RQ	0.501	0.541	0.559	2.15
CL-AP-RTQ	0.492	0.531	0.545	2.16
LC-AP-RQ	0.482	0.511	0.528	2.32
LC-AP-RTQ	0.519	0.540	0.553	2.25
LL-AP-MT	0.513	0.541	0.558	2.22
LL-AP-MTQ	0.528	0.549	0.556	2.23
LL-AP-MTRQ	**0.561**	0.582	**0.612**	**2.13**

extra quoted posts via the representation model.

Mixture Models combining Topics and Quoted Posts

In this section, to verify the effectiveness of combining given topics and quoted posts simultaneously, we introduce three types of models. Firstly, models use the representation model to integrate both texts based on the three basic hierarchical text models, namely **LL-AP-RTQ**, **CL-AP-RTQ**, **LC-AP-RTQ**. Then, the model utilizes the matching model to integrate both texts based on LL-AP, **LL-AP-MTQ**. The last model, **LL-AP-MTRQ**, incorporates given topics by the matching model and quoted posts by the representation model. Since in the former experiment, the representation model performs better to integrate quoted posts, while the matching model gains better performance in using given topics. In this experiment, we compare the mixture models with LL-AP-RQ, CL-AP-RQ, LC-AP-RQ, and LL-AP-MT as shown in Table 4.7.

Focusing on the mixture models, LL-AP-RTQ, LL-AP-MTQ, and LC-AP-RTQ show

comparable even better performance, while CL-AP-RTQ performs much worse than CL-AP-RQ. And in Table 4.6, CL-AP-RT also gains lower performance compared with CL-AP. It seems that integrating given topics with representation models will decrease the performance of the hierarchical model CL-AP. Since the representation model calculates the interaction between sentence representations from two texts. The semantics captured by the CNN model is quite different from that captured by LSTM. The inconsistency of the two representations hurt the performance. LL-AP-MTRQ, the model incorporating topics via the matching model and quoted posts via the representation model, outperforms all other models almost on all evaluation metrics. It shows the effectiveness of incorporating topics and quoted posts with the matching model and representation model respectively. As mentioned before, the representation model is suitable for quoted posts, and the matching model takes advantage of given topics. The mixture model integrating both topics and quoted posts performs better.

Extension of Topics

In previous experiments, we extend the topic description. We also run hierarchical text models on the original topic description as shown in Table 4.8. Models with * utilized the original topic descriptions. All models integrating extended topics show better performance than corresponding models integrating original topics. It proves the effectiveness of extending the descriptions of topics. Compared with all hierarchical text models, LL-AP-RT* and CL-AP-RT* perform worse, while LL-AP-MT* achieves much better performance on all four models. It shows that the matching model can make better use of topics information than the representation model. LL-AP-MT* also gains comparable even better performance than the models utilizing extended topics via the representation model including LL-AP-RT, CL-AP-RT, and LC-AP-RT. The results also prove that the matching model is much suitable to integrate topic information compared with the representation model.

Table 4.8: Experiment results of models incorporating topics given by instructors.

Model	**QWK** ↑	**SCC** ↑	**PCC** ↑	**RMSE** ↓
LL-AP-RT*	0.377	0.431	0.436	2.52
LL-AP-RT	0.444	0.482	0.504	2.34
CL-AP-RT*	0.382	0.408	0.448	2.47
CL-AP-RT	0.461	0.507	0.509	2.28
LC-AP-RT*	0.443	0.454	0.505	2.37
LC-AP-RT	0.446	0.464	0.498	2.33
LL-AP-MT*	0.485	0.485	0.518	2.29
LL-AP-MT	0.524	0.539	0.556	2.21

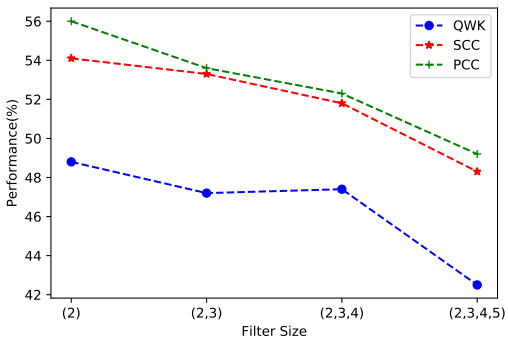
Hyper-parameter Analysis

In this section, we conduct additional experiments to show the effect of hyper-parameters. We mainly focus on the filter size of the convolutional neural network in CL-AP, CL-AP-RQ, LC-AP, LC-AP-RQ, LL-AP-MT, and LL-AP-RQMT. Since the representation model performs better to integrate quoted posts, and matching model is suitable for given topics. Then we also conduct experiments to see the influence of different learning rates in LL-AP-RQMT, the best model in our work.

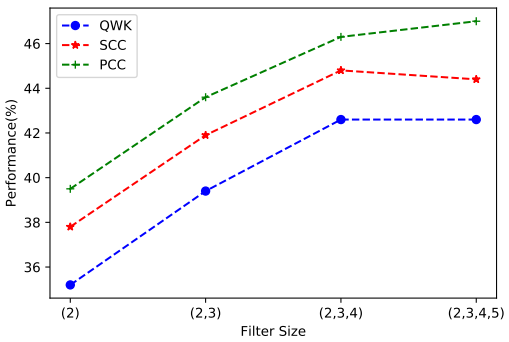
Figure 4.3(a) shows the experimental results of CL-AP with different filter sizes and evaluation metrics. The results show that with more multi-scale filters, the performance decreases. The filter sizes indicate the window size of adjacent words in each sentence. The reason may be that most key-phrases contain two words.

Figure 4.3(b) illustrates the performance of LC-AP. The filter size refers to the window size of adjacent sentences. The results show that by considering a proper number of sentences, we can obtain better text representations.

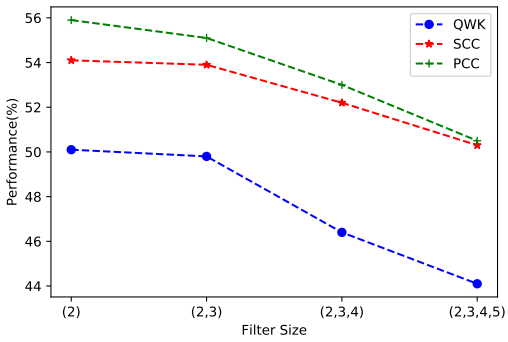
The influence of different filter sizes with different evaluation metrics of CL-AP-RQ is shown in Figure 4.3(c). The performance shows similar trends with CL-AP.



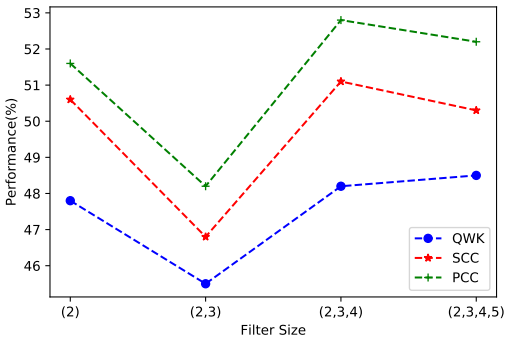
(a) CL_AP



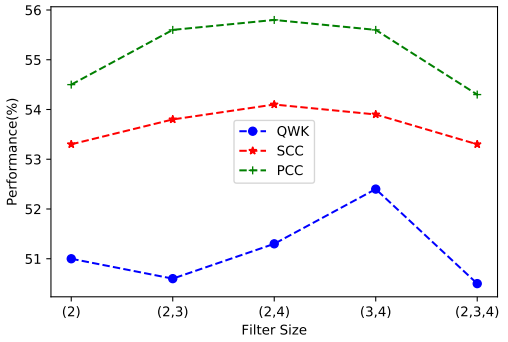
(b) LC_AP



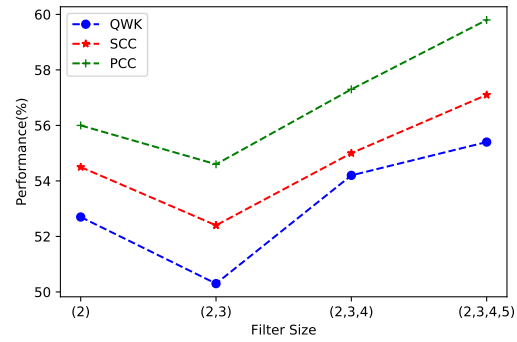
(c) CL_AP_RQ



(d) LC_AP_RQ



(e) LL_AP_MT



(f) LL_AP_MTRQ

Figure 4.3: Experimental results of various models with different filter sizes.

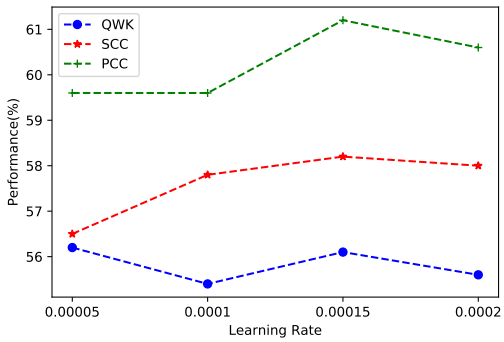


Figure 4.4: Experimental results of LL-AP-MTRQ with different learning rates.

Since using the representation model to incorporate quoted posts will not affect the representations of posts.

In Figure 4.3(d), we also show the influence of different filter sizes on LC-AP-RQ. With extra quoted posts, the filter size has more effect on the performance.

In Figure 4.3(e), the experimental results illustrate the influence of different filter sizes on the matching model, LL-AP-MT. Proper multi-scale filter sizes gain better performance.

We also show filter sizes' influence in Figure 4.3(f) of the mixture model, LL-AP-MTRQ. Since the model is more complex, multi-scale filter sizes do gain better performance.

With the illustration of the influence of different filter sizes, we also conduct experiments to see how learning rate affects the mixture model, LL-AP-MTRQ.

Figure 4.4 shows the performance of LL-AP-MTRQ with different learning rates. Slightly higher learning rates lead to better performance.

Table 4.9: Experiment results of models incorporating topics given by instructors.

Model	LL-AP	CL-AP	LC-AP	LL-AP-RQ
#Parameters	1.8M	0.96M	1.17M	1.8M
Runtime (BPS)	14.0	18.0	15.0	7.7
Model	CL-AP-RQ	LC-AP-RQ	LL-AP-MT	LL-AP-MTRQ
#Parameters	0.96M	1.17M	>1.8M	>1.8M
Runtime (BPS)	8.5	8.1	5.2	5.1

Model Parameters Analysis

In this section, we compare the model complexity measured by the number of the parameters of hierarchical text models, as well as representation models and matching models. Meanwhile, we also show the running time of different models. All the statistics are shown in Table 4.9.

We calculate the number of parameters of various models first. For the representation models, we utilize multiple attention to implement the self-attention, so these models do not require extra parameters. For the model using convolutional neural networks, the total number of the parameters highly rely on the multi-scale filter sizes. For LL-AP-MTRQ, the representation model has no extra parameters, and the matching model contains several 2-D filters. However, the total number of the parameters of the filters is significantly less than that of the LSTM. So we use ¿1.8M as the number of the parameters of the LL-AP-MTRQ model. So as LL-AP-MT. The total number of parameters of the filters is extremely small than that of LSTM.

To compare the run-time of each model, we utilize Batch Per Second (BPS) as the metric, which means that how many batches can be trained during one second. Although representation models do not require extra parameters, they still contain complicated calculations. Compared with the three hierarchical models including LL-AP, CL-AP, and LC-AP, LL-AP-RQ, CL-AP-RQ, as well as LC-AP-RQ have the same

number of parameters as the corresponding models. However, these models require more runtime.

4.5 Summary

In this paper, we propose a new task called APS to measure the writing quality and relevance of the posts with extra topics and quoted posts. To solve the proposed task, we propose a mixture model including a hierarchical text model to measure the writing quality, a semantic matching model to utilize topics, and a representation model to integrate quoted posts. Experimental results show that combining the relevance measurement via integrating topics and quoted posts can improve the performance on the correlation metrics by more than 6 percent. The model also performs better than integrating topics or quoted posts respectively. Furthermore, integrating topics via the matching model and quoted posts via the representation model achieves the best performance almost on all evaluation metrics, which proves that the semantic matching model is suitable to integrate topics and the representation model can make better use of quoted posts.

Chapter 5

Enhancing Pre-trained Models with Self-supervised Intermediate Tasks for Natural Language Understanding

5.1 Introduction

Pre-trained language models (PTMs) can be easily adapted to solve various natural language processing tasks and gain much higher performance than previous solutions. So it is a new trend to improve the performance of various tasks based on existing PTMs. An effective approach is to further train PTMs on intermediate tasks before fine-tuning them on downstream tasks.

Investigating what and how intermediate tasks benefit downstream tasks becomes a hot topic. Intermediate tasks include supervised tasks and self-supervised tasks. Most of the existing works focus on supervised tasks. For example, Pruksachatkun

et al. [73] performed a large-scale study on massive supervised intermediate and target tasks. Supervised tasks are highly dependent on labeled datasets, which is time-consuming to identify or construct. However, self-supervised tasks can be constructed more flexible and do not require human-annotated labels. The current study on self-supervised tasks only pays attention to adapting the general PTMs to specific domains. For instance, Gururangan et al. [33] proposed to utilize the masked language model (MLM) task to fine-tune PTMs on domain-specific corpora. MLM relies on extra corpora, which are labor-intensive to collect.

In this paper, we extend the study on self-supervised intermediate tasks to the general domain without requiring extra corpora. More specifically, we identify four self-supervised tasks, and then investigate how they benefit downstream tasks. The four tasks are next sentence prediction (NSP) [22], sentence ordering prediction (SOP) [53], sentence structural objective (SSO) [102], and sentence position prediction (SPP) [15]. To avoid using extra corpora, we construct new datasets from the English Wikipedia Corpus[1] which are widely used to train existing PTMs. To explore the effectiveness of identified tasks, it is necessary to conduct a fair comparison so that it is important to select a PTM that does not contain any discourse-level information. RoBERTa [60] that was trained only on the MLM task is most suitable. We fine-tune the RoBERTa model with the four tasks respectively and evaluate the fine-tuned models on three popular benchmarks namely GLUE [101], SWAG [117], and SQuAD [77] to discover in what ways the four tasks can benefit tasks in three benchmarks. Besides, with fine-tuning once, we also explore the effectiveness of multi-round fine-tuning as well as the task order. We utilize multiple tasks to fine-tune PTMs consecutively in different orders such as from easy task to hard task or in reverse. Furthermore, existing PTMs used multiple natural sentences to fulfill text segments to construct training samples. We also conduct experiments to see the effectiveness of constructed datasets with different text segments.

[1]https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

In summary, we have the following findings: i) Discourse-level information can improve the performance of tasks in the three benchmarks. ii) NSP, the task measuring the semantic relatedness, performs much better on the similarity and paraphrase tasks compared with SOP. iii) SOP, the task inferring the sentence order, achieves higher performance of inference tasks compared with NSP. iv) SSO, the combination of NSP and SOP, benefits the question answering tasks by a large margin. v) Text segments with one natural sentence perform much better than those with multiple natural sentences.

5.2 Self-supervised Intermediate Tasks

Before introducing our identified self-supervised intermediate tasks, we will briefly review the BERT model and the RoBERTa model.

5.2.1 Review of BERT and RoBERTa

BERT [22] is a multi-layer bidirectional Transformer encoder. To introduce the model specification, the number of layers (i.e., Transformer blocks) is L, the hidden size is H, and the number of self-attention heads is A. In general, there are two BERT models, namely $\mathbf{BERT_{BASE}}$ (L=12, H=768, A=12, Total Parameters=110M) and $\mathbf{BERT_{LARGE}}$ (L=24, H=1024, A=16, Total Parameters=340M). BERT utilized two pre-training tasks including masked language model and next sentence prediction, and large corpora involving the BooksCorpus (800M words) [**?**] and the English Wikipedia Corpus (2500M words)[1] to train the model.

RoBERTa [60] claimed that BERT was significantly undertrained and proposed an improved recipe for training BERT models. The modifications are: a) training the model with bigger batches, over more data; b) removing the next sentence prediction task during training; c) training on longer sequences; d) dynamically changing

the masking patterns and training the model longer. Since RoBERTa did not revise the model architect of BERT. There are also two RoBERTa models namely $\mathbf{RoBERTa_{BASE}}$ and $\mathbf{RoBERTa_{LARGE}}$.

RoBERTa has achieved much better performance than the BERT model and used only MLM as the pre-training task to capture word-level interactions. However, discourse-level information is also important. On the one hand, Devlin [22], Mandar [44], and Wang [102] have illustrated the effectiveness of discourse-level pre-training tasks. On the other hand, for downstream tasks, there are not only single-sentence classification tasks but also two-sentence similarity or inference tasks. Furthermore, referring to [36]'s idea, discourse-level information is important to understand the semantic meaning of the whole text. So it is necessary to investigate how discourse-level self-supervised intermediate tasks benefit downstream tasks.

5.2.2 Review of Self-supervised Tasks

In this section, we will introduce details of four self-supervised tasks including next sentence prediction, sentence ordering prediction, sentence structural objective, and sentence position prediction.

To implement these tasks, existing PTMs packed multiple sentences into each text segment to fulfill the whole input sequence (up to 512 tokens). However, longer text segments mean more context information that reduced the difficulty of the tasks. And the evaluation tasks consist of natural sentences so that the training process and testing process is not consistent. So we construct datasets for these tasks with natural sentences. We also conduct experiments on different text segments with multiple natural sentences.

Next Sentence Prediction

Next sentence prediction was proposed in BERT [22]. Given two sentences A and B, NSP predicts whether B is the next sentence of A. To construct the dataset, in 50% samples, B is the actual next sentence that follows A. And for the rest 50% samples, unlike the implementation in BERT which sampled B from the whole corpus, we randomly sample B in the same document that is not A or the actual next sentence to obtain more difficult negative samples.

Sentence Ordering Prediction

Sentence Ordering Prediction (SOP) was proposed in ALBERT [53]. The task aims to predict the order of two sentences (A and B). Similar to NSP, in 50% samples, two sentences are in the right order, and in the rest 50% samples, two sentences are in the reverse order.

Sentence Structural Objective

Sentence Structural Objective (SSO) was proposed in StructBERT [102]. For two sentences A and B, in $\frac{1}{3}$ samples, B is the previous sentence of A, in other $\frac{1}{3}$ samples, B is the next sentence of A, and in rest $\frac{1}{3}$ samples, B is a randomly sampled sentence in the same document that is not the previous or the next sentence of A. For two adjacent sentences A and B, the sample AB that B is the next sentence of A and sample BA that A is the previous sentence of B will both exist in the dataset and have different labels. It is similar to SOP. Also, the negative sampling is the same as NSP. So SSO is a combination of NSP and SOP.

Sentence Position Prediction

Sentence Position Prediction (SPP) is proposed by Chen [15]. Different from SOP that modeling the order of two sentences, SPP can model the order of N sentences ($N \geq 3$). With extracted N consecutive sentences from each paragraph, randomly moving one of these N sentences to the beginning. The task is to predict the true position of the first sentence. The challenge is that existing PTMs only support at most two sentences in the input. In Arman's work [20], they proposed to add a special token at the end of each sentence to construct the input of multiple sentences. So we construct new input sequences for the RoBERTa model as [<s>,S_0,</s>,S_1,</s>,...,S_N,</s>][2], and utilize the output representation of each </s> as the representation of each sentence S_i. However, the model is symmetrical so that the computation for each </s> is the same. Arman [20] proposed to set different segment id for each sentence so that the special token in each sentence has a different segment representation. For PTMs, there are only two types of segment id [0,1], since the input contains two sentences at most. So we regard the multiple input sentences as several two-sentence pairs, the segment id of the sentence sequence is set to $[0, 1, 0, 1, ..., N\%2 + 1]$. With obtained sentence representations $\{r_0, r_1, ..., r_{N-1}\}$, we utilize the features as shown in Formula 5.1 which is the same as Chen's work [15] to predict the right position of the first sentence.

$$\text{feature} = [r_0, r_0 - r_1, ..., r_0 - r_{N-1}] \tag{5.1}$$

In summary, there are three types of discourse-level self-supervised tasks referring to Lan's view [53]. The first type of task measures semantic relatedness such as NSP, a topic prediction task. Another type of task models the coherence relation of sentences including two sentences ordering (SOP) and multiple sentences ordering (SPP). They are coherence prediction tasks. The rest type of task is a mixture of tasks such as SSO.

[2]The input sequence, as well as special token, is referred to the implementation of Transformers

5.3 Experiment

5.3.1 Dataset Construction

English Wikipedia[1] has been used to train various PTMs, so we adopt it to construct datasets for NSP, SOP, SSO, and SPP respectively. The corpus is downloaded from the hyperlink1 and text content is extracted by using the script[3]. Then ScispaCy [4] is utilized to segment the sentence in each paragraph. We only select a part of the whole corpus to construct our datasets for the running memory limit. Besides, we construct datasets with different text segments for NSP, SOP, and SSO. By default, each segment contains one natural sentence. For SSP, with different total numbers of sentences, we construct two datasets namely SPP(4) and SPP(6). The statistics of constructed datasets are shown in Table 5.1. #Samples means the number of the samples in each dataset. The numbers 2 and 4 behind NSP, SOP, and SSO refer to different text segments with 2 natural sentences and 4 natural sentences.

5.3.2 Parameter Settings of Fine-tuning on Intermediate Tasks

For the GPU limit, we utilize **RoBERTa$_{\mathbf{BASE}}$** and adopt the implementation of Transformers[5]. For the tasks utilizing one sentence pairs as input such as NSP, SOP, and SSO, it is similar to the tasks in GLUE, so we set the sequence length to 128. For the tasks using two-sentence pairs as input including NSP(2), SOP(2), and SSO(2), the sequence length is set to 256, and 512 is the sequence length of the tasks consisting of four-sentence pairs, for example, NSP(4), SOP(4), and SSO(4). As for SPP, we set the sequence length to 256 for both tasks. Since our datasets are pretty large, the learning rate is set to 5E-6 and we only fine-tune RoBERTa for

[3]https://github.com/attardi/wikiextractor
[4]https://github.com/allenai/scispacy
[5]https://github.com/huggingface/transformers

Dataset	#Samples	Acc on Valid(%)
NSP (one sentence)	7.5M	97.6
NSP (two sentences)	2.7M	98.5
NSP (four sentences)	1.8M	98.3
SOP (one sentence)	7.4M	81.9
SOP (two sentences)	2.7M	87.5
SOP (four sentences)	1.8M	91.2
SSO (one sentence)	6.7M	83.5
SSO (two sentences)	2.9M	88.0
SSO (four sentences)	1.0M	88.6
SPP (four sentences, 1 epoch)	3.2M	65.3
SPP (four sentences, 3 epoch)	3.2M	70.1
SPP (six sentences, 1 epoch)	2.3M	63.2
SPP (six sentences, 3 epoch)	2.3M	64.0

Table 5.1: Statistics of constructed datasets and accuracy of the corresponding tasks on valid sets. For NSP, SOP, and SSO, 2, and 4 mean 2 or 4 natural sentences in each text segment respectively. For SPP, 4 and 6 are the numbers of sentences, and the additional 3 refers to three training epochs.

one epoch for NSP, SOP, and SSO. For SPP, we find that the task is much more difficult, and training 1 epoch gains lower accuracy on the validation set. So we train two SPP tasks for 3 epochs. We illustrate the accuracy of the validation set for all tasks on all constructed datasets as shown in Table 5.1. In this paper, we utilize +NSP, +SOP, +SSO, +SPP(4), and +SPP(6) to represent the fine-tuned models that fine-tuning on the corresponding tasks. Also, +NSP(2), +NSP(4), +SOP(2), +SOP(4), +SSO(2), +SSO(4) are the fine-tuned models that fine-tuned with the datasets containing different text segments.

Since we use the same model to run different classification tasks, the accuracy could

Table 5.2: Experimental results of five fine-tuned models on dev sets of GLUE. The result on the left and right side of character "/" for task MNLI represents MNLI-m and MNLI-mm correspondingly; F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. (4) means 4 sentences in total, and (6) refers to six sentences.

	Single Sentence		Similarity			Inference			-
Model	CoLA	SST-2	MRPC	STS-B	QQP	RTE	QNLI	MNLI-(m/mm)	Average
	8.5k	67k	3.5k	5.7k	363k	2.5k	108k	392k	-
RoBERTa	61.5	94.2	92.5	90.5	**88.9**	77.3	90.7	87.4/87.2	85.6
+NSP	**64.6**	94.2	**93.1**	91.2	88.9	80.3	90.8	87.4/87.3	**86.4**
+SOP	62.9	**94.4**	92.6	91.1	88.8	81.3	91.0	87.4/**87.4**	86.3
+SSO	62.4	**94.4**	93.0	**91.5**	88.8	**81.8**	91.0	87.4/87.2	**86.4**
+SPP (4)	62.3	94.1	92.2	91.0	88.8	78.1	**91.2**	**87.6**/87.2	85.8
+SPP (6)	62.9	94.1	92.0	91.0	88.9	77.2	91.1	87.5/**87.4**	85.8

partially reveal the difficulty of the tasks. SPP is the hardest task, and NSP is the easiest task. For two SPP tasks, training more epochs does improve their accuracy. For the rest three tasks, NSP, SOP, and SSO, larger text segments will reduce the difficulty of the task.

5.3.3 Evaluation Tasks

In this paper, we test our model on the GLUE [101], SWAG [117], SQuAD 1.0 [77], and SQuAD 2.0. We will briefly introduce each task.

General Language Understanding Evaluation (GLUE) is a widely used evaluation board to evaluate the performance of general models across a diverse set of existing Natural Language Understanding tasks that includes three types and nine tasks. Single sentence tasks include CoLA [106] and SST-2 [86]; similarity and paraphrase tasks consist of MRPC [23], STS-B [12] and QQP [6]; inference tasks involve MNLI

[6]data.quora.com/First-Quora-Dataset-Release-Question-Pairs

Table 5.3: Experimental results of all five fine-tuned models on dev sets of SWAG and SQuAD.

Model	SWAG	SQuAD 1.0		SQuAD 2.0	
		EM	F1	EM	F1
RoBERTa	82.8	82.2	89.6	77.2	80.6
+NSP	83.1	82.3	89.9	78.7	81.9
+SOP	**83.2**	82.4	89.8	78.6	82.0
+SSO	83.1	**82.8**	**90.2**	**79.1**	**82.3**
+SPP (4)	82.7	82.5	89.9	**79.1**	**82.3**
+SPP (6)	82.6	82.5	90.1	78.8	82.0

[108], QNLI constructed from Pranav' work [77], RTE [4], and WNLI constructed from [55]. Similar to BERT [22], we did not report the results on WNLI.

The Situations With Adversarial Generations (SWAG) dataset [117] was used as an evaluation task in Devlin's work [22]. It contains 113k sentence-pair completion examples evaluating grounded commonsense inference. Given a sentence, the task is to choose the most plausible continuation among the four choices.

The Stanford Question Answering Dataset (SQuAD v1.1) is a collection of 100k crowdsourced question/answer pairs [77]. Given a question and a passage from Wikipedia containing the answer, the task is to predict the answer text span in the passage.

The SQuAD 2.0 task extends the SQuAD 1.1 problem definition by allowing for the possibility that no short answer exists in the provided paragraph, making the problem more realistic.

5.3.4 Parameter Settings of Fine-tuning on Evaluation Tasks

The parameter settings include three parts for three types of datasets. For fine-tuning on the GLUE dataset, the sequence length is set to 128. And we set the learning rate and batch size to 2E-5 and 32 respectively, since our experimental results show that the RoBERTa model gained much better performance. For each task, we use 5 random seeds and fine-tune each fine-tuned model for 10 epochs. Then we select the best performance on the dev set for each seed and report the average performance. For the SWAG task, parameters recommended by Transformers[5] are used, the sequence length is set to 80, the learning rate is set to 5E-5, the batch size is 64, and we fine-tune each model for 3 epochs. As for the two SQuAD datasets, we also use the parameters provided by Transformers[5], we set the sequence length, learning rate, document stride, and batch size to 384, 3E-5, 128, and 24 respectively. Also, we fine-tune each model for 2 epochs. In this paper, we used the $\mathbf{RoBERTa_{BASE}}$ model provided by Transformers[5] so that the performance of various evaluation tasks is slightly different from that reported in the original paper. All experiments are conducted on 2 GTX 1080 Ti GPU cards.

5.3.5 Experimental Results and Analysis

In this section, we will introduce three experiments. The first experiment aims to investigate how identified four self-supervised tasks (five datasets, SPP was constructed two datasets) benefit tasks in GLUE, SWAG, and two SQuAD datasets. For NSP, SOP, and SSO, each text segment only contains one natural sentence. The second experiment attempts to explore whether multi-round fine-tuning on multiple tasks is effective. The last experiment tests the effectiveness of additional text segments with two sentences and four sentences.

Task	RoBERTa	+NSP	+SOP	+SSO
MR	89.1	89.3	89.3	**89.6**

Table 5.4: Experimental results of accuracy on dev sets of the additional sentiment classification task (MR).

Model	NSP	MRPC	STS-B	QQP
+NSP	97.6	93.1	91.2	**88.9**
+NSP*	**98.1**	93.1	**91.3**	88.8

Table 5.5: Experimental results on dev sets of the NSP task and three similarity tasks, NSP* means the newly constructed dataset that sampled negative samples from different documents.

Results on General Tasks

In this section, we test our five fine-tuned models namely +NSP, +SOP, +SSO, +SSP (4) and +SSP (6) on GLUE, SWAG, and two SQuAD datasets. The results on GLUE is shown in Table 5.3.2, and that on SWAG and SQuAD is shown in Table 5.3.2.

We discuss the experimental results on GLUE first. With fine-tuning on five self-supervised tasks, these models do gain better average performance compared with the original RoBERTa. Especially, the NSP, SOP, and SSO tasks improve the RoBERTa model by more than 0.7 percent on the GLUE dataset. And for the two SPP tasks, they also gain slightly higher performance. However, it is still a challenging task to learn multiple consequent sentence representations using PTMs. In our method, maybe the two tasks are not well implemented so that they do not perform well. All results verify that the discourse-level information is important for PTMs.

For each task in GLUE, +NSP, +SOP, and +SSO perform comparably even better. The three fine-tuned models gain larger improvement on the datasets with fewer samples. With different random seeds, tasks containing more samples gained robust performance. Perhaps large corpora of large datasets have updated the PTMs largely

so that the impact of our intermediate tasks is decreasing.

For two single sentence tasks, our models perform much better on CoLA and only gain comparable performance of SST-2. CoLA is an acceptability judgments task which is often employed to evaluate language models [54] since the predicted probabilities for a pair of minimally different sentences are directly comparable. RoBERTa is a language model trained on a large corpus. And our discourse-level tasks improve the performance of the CoLA task so that it seems that our tasks could also improve language models. As for SST-2, there are two possible reasons that our model does not perform much better. One reason is that SST is a large dataset since our models only perform comparable performance on large datasets. The other reason is that the discourse-level information is less helpful to capture sentiment information. To know the specific reason, we experiment on a small dataset, MR [68], which contains only 10K samples. The experimental results are shown in Table 5.4. Our method could slightly improve the performance, both semantic relatedness and discourse relation are necessary since +SSO obtained the highest performance on the two sentiment classification datasets.

For the three similarity and paraphrase tasks, +NSP gains better performance compared with +SOP. Since NSP is a topic prediction task, which aims to distinguish unrelated sentences. As for the three inference tasks, +SOP performs better compared with +NSP. Since sentence ordering prediction requires capturing implicit discourse relation that seems to be an inference task. SSO is the combination of NSP and SOP, so the fine-tuned model performs better on some similarity and inference tasks. In this work, we still know a few about how the combination task affects PTMs. The two SPP tasks also gain better performance of inference tasks, which shows the effectiveness of discourse relation of multiple sentences. Since the input sequence is much longer than the natural sentence in GLUE, and PTMs are not that suitable to learn multiple sentence representations. Two SPP tasks show pretty lower performance compared with the other three tasks.

For the NSP task, we sampled the sentences in the same document as negative samples, so that they may have the same topic. To eliminate the effect that the NSP task may distinguish two sentences with similar topics, we construct a new NSP dataset (NSP*) that sample the negative sentences from different documents, the experiments on the three similarity tasks are shown in Table 5.5. For the next sentence prediction task, +NSP* gains higher accuracy since the negative samples are easier. And on the three similarity tasks, +NSP* achieved comparable even higher performance. So the sentence semantic relatedness does improve similarity tasks.

For the SWAG task, +NSP, +SOP, and +SSO gain better performance which shows the effectiveness of discourse information from two natural sentences. The sequence length of the task is set to 80, so the input length of each sample from the two SPP tasks is too long. It may be one of the reasons that these two tasks show lower performance.

For the two SQuAD tasks, all tasks improve the performance on the two datasets. Especially, +SSO achieves the best performance. To solve QA tasks, locating the relevant part of the question and inferring the correct answer from the located part are two key steps [77]. The first step is similar to a semantic relatedness task, and the second step is an inference task. So both semantic relatedness and discourse information are important to solve QA tasks. The two SPP tasks also improve the performance even better than NSP and SOP. The results show that discourse relations among multiple sentences are also helpful to solve QA tasks.

Results of Multiple Tasks Fine-tuning

In this section, we investigate whether fine-tuning the PTMs many times with different intermediate tasks in different order is effective. In total, we conduct four task combinations. The first two task combinations are from easy to hard. SOP is required to identify sentence ordering which is harder than NSP that distinguishing

Table 5.6: Experimental results of multi-round fine-tuning on dev sets of part of GLUE, SWAG, and SQuAD. The evaluation metrics of tasks in GLUE are similar to Table 5.3.2.

Model	CoLA	MRPC	STS-B	RTE	QNLI	SWAG	SQuAD 1.0		SQuAD 2.0	
							EM	F1	EM	F1
+NSP	**64.6**	**93.1**	91.2	80.3	90.8	83.1	82.3	89.9	78.7	81.9
+SOP	62.9	92.6	91.1	81.3	**91.0**	**83.2**	82.4	89.8	78.6	82.0
+SSO	62.4	93.0	**91.5**	81.8	**91.0**	83.1	**82.8**	**90.2**	**79.1**	**82.3**
+NSP→SOP	62.1	92.4	91.2	**82.1**	91.0	83.2	82.2	89.7	78.7	82.1
+SOP→NSP	62.9	92.6	**91.4**	**81.9**	91.0	83.1	82.2	89.9	78.4	81.9
+NSP→SSO	63.5	93.0	91.5	81.3	90.9	82.9	82.1	89.8	78.7	82.1
+SSO→NSP	62.7	93.0	91.5	81.3	91.0	83.0	82.3	89.7	78.2	81.7

Table 5.7: Experimental results of different text segments on dev sets of part of GLUE, SWAG, and SQuAD. 2 and 4 refer to 2 natural sentences, and 4 natural sentences in each text segment respectively. The evaluation metrics of tasks in GLUE are similar to Table 5.3.2.

Model	CoLA	MRPC	STS-B	RTE	QNLI	SWAG	SQuAD 1.0		SQuAD 2.0	
							EM	F1	EM	F1
RoBERTa	61.5	92.5	90.5	77.3	90.7	82.8	82.2	89.6	77.2	80.6
BEST in Table 5.3.2, 5.3.2	**64.6**	**93.1**	**91.5**	**81.8**	**91.2**	**83.2**	**82.8**	**90.2**	**79.1**	**82.3**
+NSP(2)	61.1	92.6	91.1	79.1	90.9	82.7	82.6	89.9	78.3	81.7
+NSP(4)	61.5	92.6	91.0	78.1	90.6	82.8	82.6	90.0	77.8	81.1
+SOP(2)	61.6	92.0	90.9	80.5	90.8	83.0	82.1	89.7	78.3	81.6
+SOP(4)	62.6	91.5	91.0	80.3	90.9	82.6	82.1	89.7	78.4	81.5
+SSO(2)	61.6	92.4	91.1	80.4	91.0	82.6	82.3	89.8	78.5	81.7
+SSO(4)	62.3	91.8	91.1	79.2	90.9	82.5	82.5	89.9	78.2	81.5

unrelated sentences. We construct two tasks, the first one fine-tunes NSP first and then fine-tunes SOP shown as +NSP→SOP, the other one fine-tunes SSO followed the fine-tuning of NSP shown as +NSP→SSO. The rest two task combinations are in reverse, namely +SOP→NSP, and +SSO→NSP. For the long-running time of the large evaluation tasks, we only test four task combinations on part of GLUE including one single sentence task, two similarity tasks, and two inference tasks, SWAG, and two SQuAD tasks as shown in Table 5.3.5.

In summary, compared with fine-tuning on a single task, fine-tuning the RoBERTa model on two tasks consecutively did not improve the performance of downstream tasks significantly. However, for the two inference tasks, the additional information of sentence ordering could obtain higher performance than only fine-tuning on the NSP task. It proved that sentence ordering information is important for inference tasks. Especially for the RTE task, the multi-round fine-tuning on NSP and SOP achieved the best performance. Besides, for the STS-B task, +SOP→NSP also gained better performance than +NSP and +SOP respectively.

Observing the results of multi-round fine-tuning in different orders, there is no significant difference except that on CoLA and SQuAD 2.0. For the CoLA task, different task combinations rely on different task orders. The two combinations of NSP and SOP perform lower performance than +NSP and +SOP respectively. As for the SQuAD 2.0 task, the task order seems to be much important. For four task combinations, the two combinations namely NSP→SOP and NSP→SSO that the task order is from easy to hard achieved much better performance than the rest two task combinations that the task order is in reverse.

Results of Different Text Segments

In this section, to verify the impact of different text segments, we construct additional two types of text segments for NSP, SOP, and SSO, namely two natural sentences,

and four natural sentences. We illustrate the experiment results on part of GLUE including a single sentence task, two similarity tasks, and two inference tasks, SWAG, and two SQuAD tasks as shown in Table 5.3.5.

In summary, fine-tuning the RoBERTa model on tasks with larger text segments did not gain better performance than that with the text segment consisting of one natural sentence. Since the tasks in GLUE involve samples of natural sentences, larger segments lead to inconsistency between training and testing.

For the CoLA task, it is interesting that the text segment with four natural sentences leads to better performance than that with two natural sentences. For the tasks of MRPC and RTE, larger text segments result in lower performance. As for the STS-B and QNLI tasks, the results of different text segments show little difference. Another interesting observation is that on the SQuAD 1.0 task, the NSP task with larger text segments leads to better performance.

5.4 Summary

In this paper, we extend the study on self-supervised intermediate tasks to the general domain and other self-supervised tasks. We identify what and how these tasks can benefit downstream tasks. We also observe some interesting findings through the experiments.

In another view, we propose a new way to use pre-trained language models, which is to design specific self-supervised tasks for specific target tasks. Designing new self-supervised tasks for focused target tasks is one of our future directions.

Chapter 6

Enhancing Automated Essay Scoring Performance via Fine-tuning Pre-trained Language Models with Combination of Regression and Ranking

6.1 Introduction

Automated Essay Scoring (AES) automatically evaluates the writing quality of essays. Essay assignments evaluation costs lots of time. Besides, the same instructor scoring the same essay at different times may assign different scores (intra-rater variation), different raters scoring the same essay may assign different scores (inter-rater variation) [85]. To alleviate teachers' burden and avoid intra-rater variation, as well as inter-rater variation, AES is necessary and essential. An early AES system, e-rater [17], has been used to score TOEFL writings.

Recently, large pre-trained language models, such as GPT [74], BERT [22], XLNet [113], etc. have shown the extraordinary ability of representation and generalization. These models have gained better performance in lots of downstream tasks such as text classification and regression. There are many new approaches to fine-tune pre-trained language models. Sun et al. [90] proposed to construct an auxiliary sentence to solve aspect-based sentiment classification tasks. Cohan et al. [20] added extra separate tokens to obtain representations of each sentence to solve sequential sentence classification tasks. Sun et al. [91] summarized several fine-tuning methods, including fusing text representations from different layers, utilizing multi-task learning, etc. To our knowledge, there are no existing works to improve AES tasks with pre-trained language models. Before introducing our new way to use pre-trained language models, we briefly review existing works in AES firstly.

Existing works utilize different methods to learn text representations and constrain scores, which are the two key steps in AES models. For text representation learning, various neural networks are used to learn essay representations, such as Recurrent Neural Network (RNN) [92, 93], Convolutional Neural Network (CNN) [92], Recurrent Convolutional Neural Network (RCNN) [24], etc. However, simple neural networks like RNN and CNN focus on word-level information, which is difficult to capture word connections in long-distance dependency.

Besides, shallow neural networks trained on a small volume of labeled data are hard to learn deep semantics. As for score constraints, prediction and ranking are two popular solutions. From the prediction perspective, the task is a regression or classification problem [92, 93, 24]. Besides, from the recommendation perspective, learning-to-rank methods [114, 14] aim to rank all essays in the same order as that ranked by gold scores. However, without precise score mapping functions, only regression constraints could not ensure the right ranking order. And only ranking based models could not guarantee accurate scores. In general, there are two key challenges for the AES task. One is how to learn better essay representations to evaluate the writing quality, the

other one is how to learn a more accurate score mapping function.

Motivated by the great success of pre-trained language models such as BERT in learning text representations with deep semantics, it is reasonable to utilize BERT to learn essay representations. Since self-attention is a key component of the BERT model, it can capture the interactions between any two words in the whole essays (long texts). Previous work [91] shows that fusing text representations from different layers does not improve the performance effectively. For the AES task, the length of essays approximates the length limit of the BERT model, so it is hard to construct an auxiliary sentence. Meanwhile, only score labels are available; it is also difficult to utilize multi-task learning. Summarized existing works in AES, they utilize regression loss or ranking loss, respectively. Regression loss requires to obtain accurate score value, and ranking loss aims to get precise score order. Unlike multi-task learning requires different fully-connected networks for different tasks, we propose to constrain the same task with multiple losses to fine-tune the BERT model.

In addition, it is impossible to rank all essays in one batch so that the model is required to learn more accurate scores. During training, the weight of the regression loss is increasing while that of ranking loss is decreasing.

In this paper, we propose R^2BERT (BERT Model with Regression and Ranking). In our model, BERT is used to learn text representations to capture deep semantics. Then a fully connected neural network is used to map the representations to scores. Finally, regression loss and batch-wise ranking loss constrain the scores together, which are jointly optimized with dynamic combination weights.

To evaluate our model, an open dataset, Automated Student Assessment Prize (ASAP), is used. With the measurement of Quadratic Weighted Kappa (QWK), our model outperforms state-of-the-art neural models on average QWK score of all eight prompts near 3 percent and also performs better than the latest statistical model. Especially on the two narrative Prompts (7 and 8), only the regression based model performs

comparably even better compared with other models. And our model with combined loss gains much better performance. To explain the model's effectiveness, we also illustrate the attention weights on two example essays (an argumentative essay and a narrative essay). The self-attention can capture most conjunction words that reveal the logical structure, and most key concepts that show the topic shifting of the narratives.

In summary, our contributions are:

- We propose a new method called multi-loss to fine-tune BERT models in AES tasks. We are also the first one to combine regression and ranking in these tasks. The experiment results show that the combined loss could improve the performance significantly.
- Experiment results also show that our model achieves the best average QWK score and outperforms other state-of-the-art neural models almost on each prompt.
- To show the effectiveness of self-attention in the BERT model, we illustrate the weights of different words on two examples, including one argumentative essay and one narrative essay.

6.2 R^{2}BERT

In this section, we first introduce the framework of our model, briefly review the BERT model, as well as self-attention. In addition, we will illustrate the regression model as well as some useful tricks. Finally, we will show batch-wise learning to rank model and the combination metric.

Our model, as shown in Figure 6.1, takes a batch of essays as input. With preprocessing (adding a special token, [CLS], at the beginning of each essay), each token

is transformed into its embedding and sent into the BERT model. The representations of all essays are the output vectors mapping to [CLS]. Essay scores could be obtained by passing the representations into the Score Mapping Function. They are constrained by regression loss and ranking loss, which are optimized jointly with the dynamic combination. As shown in the color bar, the weight of regression loss is gradually increasing, while that of ranking loss is decreasing.

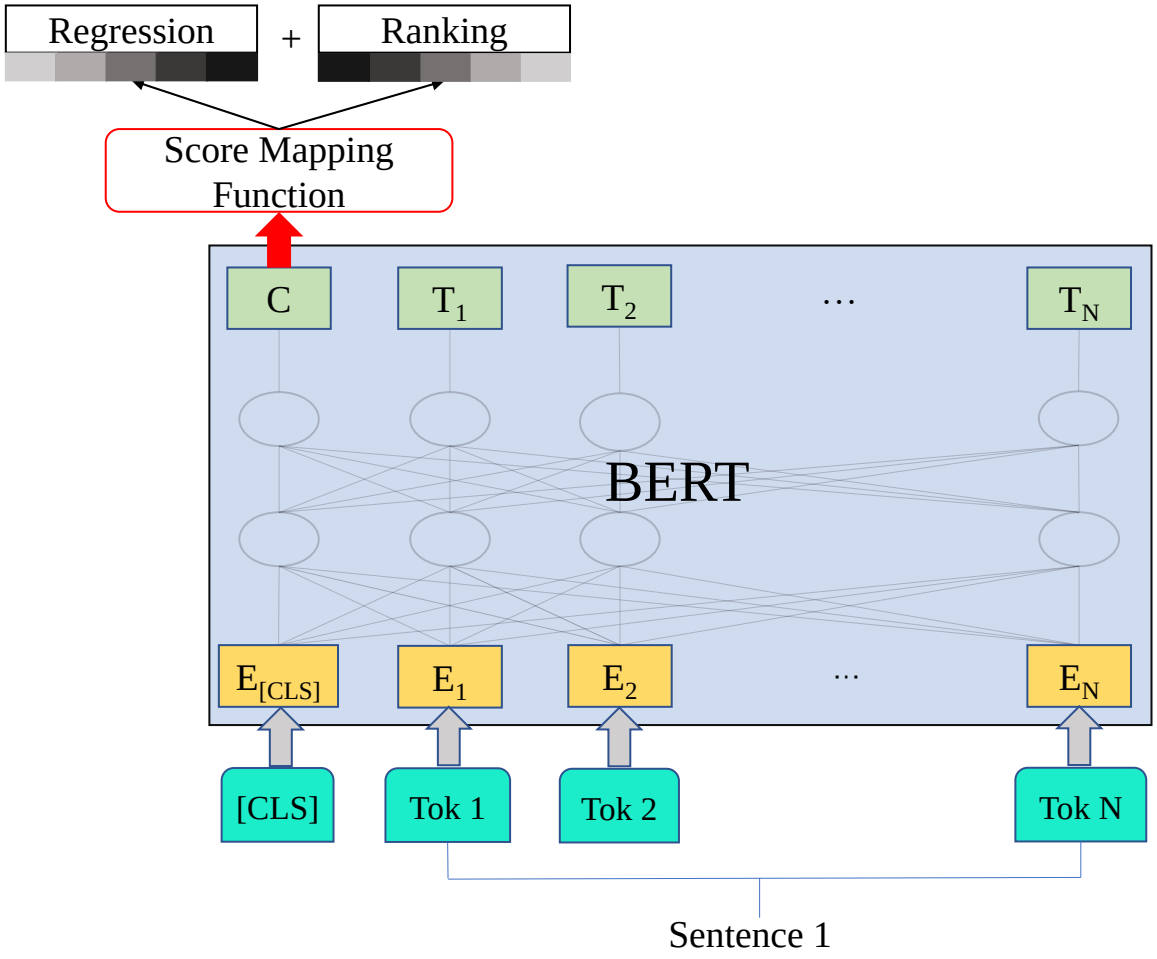


Figure 6.1: R^{2}BERT Framework

6.2.1 BERT

BERT [22] refers to Bidirectional Encoder Representations from Transformers, which is one of the most popular models in recent years. More specifically, BERT is an extremely large pre-trained language model, which was trained on enormous corpora, totally more than 3000M words. Meanwhile, two target tasks, namely masked language model and next sentence prediction, are used to train the model. Many downstream tasks of natural language processing have gained benefits by utilizing pre-trained BERT to get text representation such as sentence classification, question answer, common sense inference, etc. To benefit regression problems, the target task is replaced by a fully connected neural network. Then the whole BERT model is fine-tuned on the new dataset.

Generally BERT has two parameter intensive settings:

BERT$_{\texttt{BASE}}$: 12 layers, 768 hidden dimensions and 12 attention heads (in transformer) with the total number of parameters, 110M;

BERT$_{\texttt{LARGE}}$: 24 layers, 1024 hidden dimensions and 16 attention heads (in transformer) with the total number of parameters, 340M.

6.2.2 Self-attention

Self-attention [98] is the key to the success of BERT, which is a mechanism that a sequence calculates the word weights with itself. Given a text, we construct a matrix W with three copies Q, K, V, referring to query, key, and value, in which each column is the word embedding. The new words' representations are calculated via the attention as shown in Formula 6.1, and Formula 6.2, where d is the size of word embedding, n_Q, n_K and n_V denote the number of words in each text, $Q[i]$ is the i_{th} word representation in the query text Q.

$$\text{Att}(Q, K) = [\text{softmax}(\frac{Q[i] \cdot K^T}{\sqrt{d}})]_{i=0}^{n_Q - 1} \tag{6.1}$$

$$V_{att}(Q, K, V) = \text{Att}(Q, K) \cdot V \in R^{n_Q \times d} \tag{6.2}$$

6.2.3 Feature Extraction

Given a sample essay $t = \{w_1, w_2, .., w_N\}$ as input, where N is the number of the words, we preprocess it to a new sequence $t' = \{[\text{CLS}], w_1, w_2, .., w_N\}$, where [CLS] is a special token. Assuming BERT$(\cdot)$ is the pre-trained BERT model, we can obtain the hidden representations of all the input words, $h = \text{BERT}(t') \in R^{r_h * |t'|}$, where $|t'|$ is the length of the input sequence and r_h is the dimension of the hidden state. Finally, the hidden representation mapping to [CLS], $r = h_{[\text{CLS}]}$, is used as the text representation.

6.2.4 Regression

With obtained text representation r, a fully connected neural network FCNN$(\cdot)$ is used as the score mapping function. More specifically, FCNN is a linear combination function, where $\boldsymbol{W}$ is the weight matrix and $\boldsymbol{b}$ is the bias as shown in Formula 6.3. To learn better parameters, the mean score of all training essays is used to initialize the bias $\boldsymbol{b}$. In addition, $\sigma = \text{Sigmoid}(\cdot)$, a non-linear activation function is used to normalize the calculated score into $[0, 1]$ as shown in Formula 6.4.

$$\text{FCNN}(r) = \boldsymbol{W}r + \boldsymbol{b} \tag{6.3}$$

$$s' = \sigma(\text{FCNN}(r)) \tag{6.4}$$

Mean square error is a widely used loss function for regression tasks. Given a dataset $D = \{(t_i, s_i) | i \in [1 : m]\}$, m is the number of samples, and t_i refers to the i_{th} essay.

Besides, s_i is the gold score of the i_{th} essay. The regression objective L_m is shown in Formula 6.5.

$$L_m = \text{MSE}(s, s') = \frac{1}{m}\sum_{i=1}^{m}(s_i - s'_i)^2 \tag{6.5}$$

6.2.5 Batchwise Learning to Rank Model

ListNet [11] ranks a list of objectives each time and measures the accordance between the predicted ranking list and the ground truth label. In our problem, all the essays are a large list. However, it is impossible to rank all the essays in one batch. We sacrifice the accuracy and only rank essays in each batch, which we called batch-wise ListNet.

Before introducing the objective of ListNet, we will give several basic definitions. Suppose that given a set of essays which are identified with the numbers $\{1, 2, ..., m\}$. A permutation π on the essays is defined as a bijection from $\{1, 2, ..., m\}$ to itself. The permutation is written as $\pi =< \pi(1), \pi(2), ..., \pi(m) >$, where $\pi(i)$ refers to the essay at position i in the permutation. And we also assume any permutation is possible. The set of all possible permutations is denoted as Ω_m. As aforementioned, we assume the batch size is m, and the calculated score of the essay pointed by $\pi(i)$ is $s'_{\pi(i)}$. As given by the original paper [11], the permutation probability is defined as Formula 6.6. And $\Phi(\cdot)$ is an increasing and strictly positive function.

$$P_{s'}(\pi) = \prod_{j=1}^{n}\frac{\Phi(s'_{\pi(j)})}{\sum_{k=j}^{n}\Phi(s'_{\pi(k)})} \tag{6.6}$$

The top one probability $P_{s'}(j)$ is defined as Formula 6.7, where j refers to each essay in the batch.

$$P_{s'}(j) = \frac{\Phi(s'_j)}{\sum_{k=1}^{n}\Phi(s'_k)} \tag{6.7}$$

With the use of top one probability, given two lists of scores s and s' as aforementioned, Cross Entropy could be used to represent the distance (batchwise loss function L_r)

between the two score lists as shown in Formula 6.8.

$$L_r = \texttt{CE}(s, s') = -\sum_{j=1}^{n} P_s(j) \log(P_{s'}(j)) \tag{6.8}$$

6.2.6 Combination of Regression and Ranking

The key problem of loss combination is to determine the weight of each loss. In the scoring scenario, teachers always prefer to score each essay rather than ranking all the essays. Besides, using batch-wise learning to rank approach could not guarantee precise global order. Referring to the combination method proposed in [**?**], the weight of ranking loss is decreasing, and that of regression loss is increasing during training. The weight calculation is followed by Formula 6.9, where τ_e is a σ function about e calculated as Formula 6.10.

$$L = \tau_e \times L_m + (1 - \tau_e) \times L_r \tag{6.9}$$

$$\tau_e = \frac{1}{1 + \exp(\gamma(E/2 - e))} \tag{6.10}$$

In Formula 6.10, E is the total number of the epochs, and e is the value of current epoch, γ is a hyper-parameter which is chosen such that $\tau_1 = 0.000001$.

6.3 Experiment

In this section, the ASAP dataset is introduced firstly. Then we illustrate experiment settings and evaluation metrics. In addition, baseline models, experiment results, and analyses are shown. Furthermore, we also visualize the attention weights of different words in two examples.

Set	#Essays	Genre	Avg Len.	Range
1	1783	ARG	350	2-12
2	1800	ARG	350	1-6
3	1726	RES	150	0-3
4	1772	RES	150	0-3
5	1805	RES	150	0-4
6	1800	RES	150	0-4
7	1569	NAR	250	0-30
8	723	NAR	650	0-60

Table 6.1: Statistics of the ASAP dataset; Range means the score range, For genre, ARG, RES, and NAR map to argumentative essays, response essays and narrative essays respectively.

6.3.1 Dataset

The automated Student Assessment Prize dataset comes from a Kaggle competition[1], which contained eight essay prompts with different genres, including argumentative essays, response essays, and narrative essays. Each essay was given a score by the instructors. Some statistical information is shown in Table 6.1.

6.3.2 Experiment Settings

Following previous work, we also utilize 5-fold cross-validation to evaluate all models with 60/20/20 split for train, validation, and test sets, which are provided by [92]. We implement our model based on the Pytorch implementation of BERT [2] and use the $\text{BERT}_{\texttt{BASE}}$ model due to the limit of GPU memory. Besides, we truncate all the essays with the max length of 512 words, following the setting of BERT. Also, for the limit

[1]https://www.kaggle.com/c/asap-aes/data

[2]https://github.com/huggingface/pytorch-transformers

Table 6.2: QWK evaluation scores on ASAP dataset (* means statistical model)

		Dataset/Prompts								
ID	Model	1	2	3	4	5	6	7	8	Average
1	LSTM(last)	0.165	0.215	0.231	0.436	0.381	0.299	0.323	0.149	0.275
2	BiLSTM(last)	0.226	0.276	0.239	0.502	0.375	0.412	0.361	0.188	0.322
3	LSTM(mean)	0.582	0.517	0.516	0.702	0.604	0.670	0.661	0.566	0.602
4	BiLSTM(mean)	0.591	0.491	0.498	0.702	0.643	0.692	0.683	0.563	0.608
5	*EASE(SVR)	0.781	0.630	0.621	0.749	0.782	0.771	0.727	0.534	0.699
6	*EASE(BLRR)	0.761	0.621	0.606	0.742	0.784	0.775	0.730	0.617	0.705
7	CNN+LSTM	0.821	0.688	0.694	0.805	0.807	0.819	0.808	0.644	0.761
8	LSTM-CNN-att	0.822	0.682	0.672	0.814	0.803	0.811	0.801	0.705	0.764
9	RL1	0.766	0.659	0.688	0.778	0.805	0.791	0.760	0.545	0.724
10	SKIPFlOW	0.832	0.684	0.695	0.788	0.815	0.810	0.800	0.697	0.764
11	*HISK+BOSWE	**0.845**	**0.729**	0.684	0.829	0.833	0.830	0.804	0.729	0.785
12	RankingOnly	0.791	0.687	0.665	0.811	0.797	0.821	0.821	0.651	0.756
13	RegressionOnly	0.800	0.679	0.679	0.822	0.803	0.797	0.837	0.725	0.768
14	R^2BERT	0.817	0.719	**0.698**	**0.845**	**0.841**	**0.847**	**0.839**	**0.744**	**0.794**

of our GPU memory, the batch size is set to 16. Since essays in the ASAP dataset is much longer than that in GLUE [101], we fine-tune our model for 30 epochs and select the best model based on the performance on the validation set. We adjust the fine-tuning learning rate from 1e-5 to 9e-5 with the step 1e-5, and 4e-5 performs best. And γ in Formula 6.10 is set to 0.99999. For tokenization and vocabulary, we all use the pre-processing tools provided by the BERT model. We also normalize all score ranges to within [0,1]. All the scores are rescaled back to the original prompt-specific scale for calculating Quadratic Weighted Kappa scores. Following previous works, we conduct the evaluation in prompt-specific fashion.

6.3.3 Evaluation Metric

Following previous works, Quadratic Weighted Kappa (QWK) is used as the evaluation metric, which measures the agreement between calculated scores and gold ones.

To calculate QWK, a weight matrix W is constructed firstly, as shown in Formula 6.11, where i and j are gold scores and calculated scores respectively, and N is the number of possible ratings.

$$W_{i,j} = \frac{(i-j)^2}{(N-1)^2} \tag{6.11}$$

In addition, a matrix O is calculated, such that $O_{i,j}$ denotes the number of essays obtained a rating i by the human annotator and a rating j by the AES system. Another matrix E with the expected count is calculated as the outer product of histogram vectors of the two ratings. The matrix E is then normalized such that the sum of elements in E is the same as that of elements in O. Finally, with given matrices O and E, the QWK score is calculated according to Formula 6.12.

$$\kappa = 1 - \frac{\sum_{i,j} W_{i,j} O_{i,j}}{\sum_{i,j} W_{i,j} E_{i,j}} \tag{6.12}$$

6.3.4 Baselines and Implementation Details

In this section, we list several baseline models as well as state-of-the-art models.

- ***EASE** A statistical model called Enhanced AI Scoring Engine (EASE) is an AES system that is publicly available, open-source[3], and also achieved excellent results in the ASAP competition. EASE utilizes hand-crafted features such as length-based features, POS tags, and word overlap, as well as different regression techniques. Following previous works, we report the results of EASE with the settings of Support Vector Regression (SVR) and Bayesian Linear Ridge Regression (BLRR).

[3]https://github.com/edx/ease

- **LSTM** We use two layers LSTM and biLSTM, as well as mean pooling and last output to obtain the essay representations. Mean pooling means the average vector of all the hidden states, while the last output refers to the last hidden state. Then, a fully connected linear layer, as well as σ activation function, is used to gain scores. In these four models, GloVe [70] is used to initialize the word embedding matrix, and the dimension is 300.

- **CNN+LSTM** This model is proposed in Kaveh's work [92], which assembled CNN and LSTM to gain scores. We use the performance reported in the paper.

- **LSTM-CNN-att** Dong [24] proposed to use attention mechanisms and hierarchical neural networks to learn the representation of the essays. We also use the experiment results reported in their paper.

- **RL1** Wang [104] proposed a reinforcement learning based model. In that paper, QWK is used as the reward function, and classification is used to gain the scores. The performance reported in the paper is used.

- **SKIPFlOW** Yi [93] proposed the model, which considered the coherence when learning text representations. Experiment results in the paper are used.

- ***HISK+BOSWE** Cozma [21] proposed another statistical model. It utilized string kernel and word embedding to extract text features and gained higher performance.

Our models We not only show the performance of R^2BERT but also the results of the regression only version (RegressionOnly) and the ranking only version (RankingOnly). All experiments are conducted on a Linux machine running a single Tesla P40 GPU.

ID	Model	First 512	Last 512
1	RankingOnly	0.657	0.644
2	RegressionOnly	0.724	0.725
3	R^2BERT	0.743	0.745

Table 6.3: QWK evaluation scores on Prompt 8 of ASAP Dataset with different parts of the whole essays

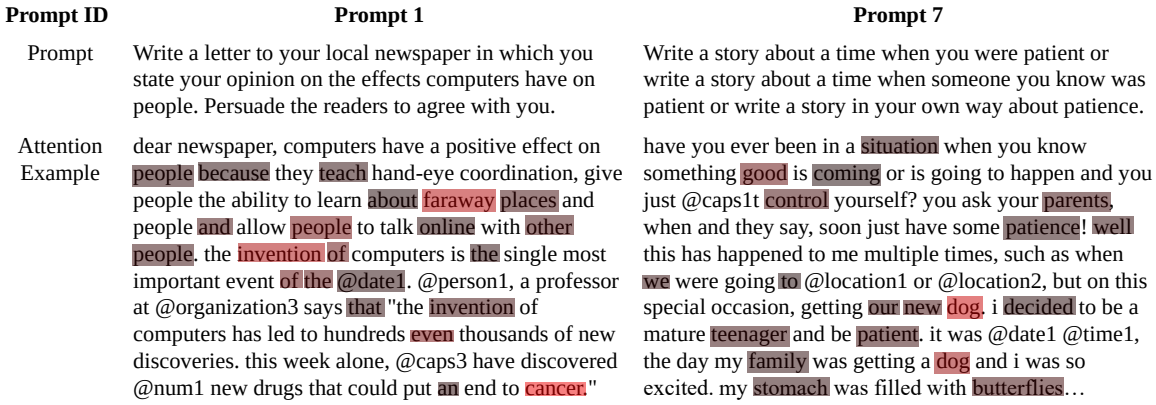


Figure 6.2: Self-attention visualization on examples of Prompt 1 and 7

6.3.5 Experiment Results and Analysis

Table 6.2 shows the empirical results of all deep learning models as well as the statistical models. First, the comparison between LSTM based models is discussed. The mean pooling performs better than the last output in all LSTM based models. Since essays in the dataset contain hundreds of words, it is difficult for LSTM to capture longer dependency. Compared with the last output, average pooling could alleviate the aforementioned problem. Meanwhile, bidirectional LSTM based models perform comparably even better than the unidirectional ones. Because the bidirectional models could capture complete context information. However, these models show lower performance than that of EASE. It means well-designed hand-crafted features are

more effective than simple neural networks. These models still perform worse than state-of-the-art models.

Additionally, we firstly compare published state-of-the-art results. RL1 [104], the reinforcement learning based model, shows pretty lower performance in recent works. Since it utilizes dilated LSTM to learn essay representations, which ignores sentence-level structure information. It still outperforms basic LSTM based models, which shows the effectiveness of the QWK reward function. CNN+LSTM [92] is an ensemble model that shows comparable performance compared with LSTM-CNN-att [24], the hierarchical model, on Prompt 1,2,4,5,6,7, and even gains much higher performance on Prompt 3. Both models outperform LSTM based models. It means that the ensemble model could make up shortages of single neural networks and performs comparably with hierarchical models. Besides, LSTM-CNN-att and SKIPFLOW [93] both are hierarchical models. They capture the explicit structure through modeling the relationship of adjacent sentences (semantics) in each essay. So they perform better in Prompt 1 and 2, which contain argumentative essays. Especially the SKIPFLOW model even gains much better performance on Prompt 1. LSTM-CNN-att also performs better on Prompt 8. However, a well-designed statistical model, HISK+BOSWE [21], outperforms all previous neural models, which also performs best on the two argumentative prompts.

Compared with previous state-of-the-art neural models, the RegressionOnly model outperforms all other neural models on the average QWK score, which shows the great power of the pre-trained language model (BERT) in capturing deeply semantic information. Especially on the two narrative prompts (Prompt 7 and 8), the RegressionOnly model outperforms other models by a large margin, which shows that self-attention is more suitable for narrative essays since it can capture key concepts in narrative essays as shown in Figure 6.2. RankingOnly model shows much lower performance on Prompt 8 as well as average QWK score, maybe because it is difficult to utilize batch-wise order to reconstruct the global order perfectly.

R^2BERT outperforms RegressionOnly and RankingOnly models on each prompt by a large margin except Prompt 7. The result means that ranking and regression are surely two complementary objectives, and a combination via dynamic weights could improve the performance effectively. In general, R^2BERT gains a much higher average QWK score compared with the aforementioned neural models and almost performs best on each prompt except Prompt 1. It illustrates a successful way to enhance BERT on downstream tasks. Only utilizing BERT to learn text representations is not enough. Suitable auxiliary objectives are also necessary. More importantly, our model also outperforms HISK+BOSWE, the latest statistical model, which proves the great power of neural networks.

BERT limits the length of each input text with a maximum of 512 words. In Prompt 8, the average length of all essays is about 650 words, which is larger than the limit. We use the first 512 words or the last 512 words instead of the whole essay. Table 6.3 shows the experimental results. Our three models achieve similar performance. How to fully use the whole essays with BERT is a direction in future works. In Table 6.2, we use the average performance as the result of Prompt 8 in each model.

In Figure 6.2, we visualize the word weights of self-attention of two essays, including an argumentative essay from Prompt 1 and a narrative essay from Prompt 7. For the limit of the page, we only demonstrate part of each essay. In the figure, the word in darker red gains lower attention weight. The argumentative example needs to convince people that computers can benefit our life. Self-attention has identified several connectors such as "because", "and", "even", and some words indicating arguments including "about", "that" etc. These words show the explicit logical structure of argumentative essays. The narrative example uses the example of getting a dog to show his/her patience. Self-attention capture the story details such as "dog", "parent", "family", "stomach", "butterflies", as well as the topic words "patient" and "patience". All these words show the topics shifting of narratives.

Model	TR	IPS	#Param
LSTM	2m53s	0.0013s	1.4M
BiLSTM	3m15s	0.0014s	1.4M
R^2BERT	22m20s	0.9103s	110M

Table 6.4: Comparison of Runtime and Memory. TR means the total training runtime on the train set and IPS means inference runtime per each test sample. #Param refers to the number of parameters.

6.3.6 Runtime and Memory

In this section, we analyze the runtime and memory, which means the total number of parameters. Since little previous work provided the source code so that it is difficult to estimate the total number of parameters accurately. Our three models only utilize different losses, so they have the same number of parameters. In summary, we only compare LSTM, BiLSTM, and R^2BERT model. Firstly, we estimate the total number of parameters for the three models. Then we record the total training time on all training samples in Prompt 6. Since simple neural networks need more training epochs to converge, yet BERT model only needs less training epochs to fine-tune. To compare the inference time, we record the time for inference per sample. All results are shown in table 6.4. It is obvious that BERT has more parameters and spends much more training and inference time. However, the inference time of each sample is near 1 second, which is practical in the real educational scenarios.

6.4 Summary

From experimental results, we can obtain several conclusions: 1) BERT is a significantly effective model to improve the performance of downstream natural language processing tasks. 2) Regression loss and ranking loss are two complementary losses.

3) Simply fine-tuning on BERT is not enough. Multi-loss objective is an effective approach to fine-tune the BERT model. 4) Self-attention is useful to capture conjunction words and key concepts in essays. In the future, we will investigate how to utilize the whole long text with the pre-trained BERT model.

Chapter 7

Conclusions and Future Directions

In this thesis, we mainly focus on post-processing and applications of pre-trained models for natural language processing. More specifically, we proposed different methods to enhance different types of pre-trained models. Meanwhile, we also proposed different approaches to utilize pre-trained models to obtain higher performance.

In Chapter 3, we proposed a novel approach to utilize glossary to enhance pre-trained word embedding models so that these models can capture more topical and functional information. To apply pre-trained word embedding models to text assessment tasks, we proposed a new task named Automated Post Scoring in Chapter 4. We proposed a representation model and matching model to integrate given topics and quoted posts and enhanced the prediction of students' posts. In Chapter 5, we proposed to use self-supervised intermediate tasks to improve pre-trained language models. Furthermore, we investigated how self-supervised intermediate tasks benefit various downstream tasks. Finally, we proposed a new method to enhance pre-trained models on Automated Essay scoring in Chapter 6. Specifically, we combined regression loss and ranking loss together to get better performance.

Reviewing the development of pre-trained models, word2vec was proposed in 2013 [63], and pre-trained langauge models were propsoed in 2018 [74]. BERT [22] started

the era of pre-trained language model. Although pre-trained models have gained success in various downstream tasks. There are still many challenge issues. The first issue is how to enhance these models on specific tasks. All existing pre-trained models are trained on general domain however, general models don't mean best models on specific tasks. The second issue is the robustness of these models, various works found that slightly changes of the input will lead to quiet different results. In addition, large pre-trained models are also sensitive to hyper-parameters. In our experiments, we found that improper parameters will lead to hard convergence of these models. Tacking these challenges will promote the research in pre-trained language models and benefit more downstream tasks.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[2] Collin F Baker, Charles J Fillmore, and John B Lowe. The berkeley framenet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, 1998.

[3] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

[4] Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

[5] Jiang Bian, Bin Gao, and Tie-Yan Liu. Knowledge-powered deep learning for word embedding. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 132–148. Springer, 2014.

[6] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[7] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[8] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[9] Tom Bosc and Pascal Vincent. Auto-encoding dictionary definitions into consistent word embeddings. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1522–1532. Association for Computational Linguistics, 2018.

[10] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47, 2014.

[11] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[12] Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, 2017.

[13] Dhivya Chandrasekaran and Vijay Mago. Evolution of semantic similarity—a survey. *ACM Computing Surveys (CSUR)*, 54(2):1–37, 2021.

[14] Hongbo Chen and Ben He. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1752, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[15] Mingda Chen, Zewei Chu, and Kevin Gimpel. Evaluation benchmarks and learning criteria for discourse-aware sentence representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 649–662, 2019.

[16] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[17] Martin Chodorow and Jill Burstein. Beyond essay length: evaluating e-rater®'s performance on toefl® essays. *ETS Research Report Series*, 2004(1):i–38, 2004.

[18] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[19] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, 2019.

[20] Arman Cohan, Iz Beltagy, Daniel King, Bhavana Dalvi, and Daniel S Weld. Pretrained language models for sequential sentence classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3684–3690, 2019.

[21] Mădălina Cozma, Andrei Butnaru, and Radu Tudor Ionescu. Automated essay scoring with string kernels and word embeddings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2:*

Short Papers), pages 503–509, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[23] William B Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[24] Fei Dong, Yue Zhang, and Jie Yang. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162, Vancouver, Canada, August 2017. Association for Computational Linguistics.

[25] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 13063–13075, 2019.

[26] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. All nlp tasks are generation tasks: A general pretraining framework. *arXiv preprint arXiv:2103.10360*, 2021.

[27] Scott Elliot. Intellimetric: From here to validity. *Automated essay scoring: A cross-disciplinary perspective*, pages 71–86, 2003.

[28] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H Hovy, and Noah A Smith. Retrofitting word vectors to semantic lexicons. In *HLT-NAACL*, 2015.

[29] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. *ACM Transactions on information systems*, 20(1):116–131, 2002.

[30] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. Ppdb: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, 2013.

[31] Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart, and Anna Korhonen. Simverb-3500: A large-scale evaluation set of verb similarity. *arXiv preprint arXiv:1608.00869*, 2016.

[32] Goran Glavaš and Ivan Vulić. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45. Association for Computational Linguistics, 2018.

[33] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics.

[34] Project Gutenberg. The online plain text english dictionary. `http://www.mso.anu.edu.au/~ralph/OPTED/`, 2009.

[35] Zellig Harris et al. Distributional hypothesis. *Word World*, 10(23):146–162, 1954.

[36] Zellig S Harris. Discourse analysis. In *Papers on syntax*, pages 107–142. Springer, 1981.

[37] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[38] Michael Heilman and Nitin Madnani. ETS: domain adaptation and stacking for short answer scoring. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 275–279, 2013.

[39] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695, 2015.

[40] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[41] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339. Association for Computational Linguistics, 2018.

[42] Sergio Jiménez, Claudia Jeanneth Becerra, and Alexander F. Gelbukh. SOFTCARDINALITY: hierarchical text overlap for student response analysis. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 280–284, 2013.

[43] Hwiyeol Jo and Stanley Jungkyu Choi. Extrofitting: Enriching word representation and its vector space with semantic lexicons. *arXiv preprint arXiv:1804.07946*, 2018.

[44] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77, 2020.

[45] N Kalchbrenner, E Grefenstette, and Philip Blunsom. A convolutional neural network for modelling sentences. In *52nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2014.

[46] Zixuan Ke and Vincent Ng. Automated essay scoring: a survey of the state of the art. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6300–6308. AAAI Press, 2019.

[47] Yoon Kim. Convolutional neural networks for sentence classification. emnlp, 2014.

[48] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[49] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[51] Sachin Kumar, Soumen Chakrabarti, and Shourya Roy. Earth mover's distance pooling over siamese lstms for automatic short answer grading. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2046–2052, 2017.

[52] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics.

[53] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[54] Jey Han Lau, Alexander Clark, and Shalom Lappin. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive science*, 41(5):1202–1241, 2017.

[55] Hector Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*. Citeseer, 2012.

[56] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, 2014.

[57] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

[58] Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1501–1511, 2015.

[59] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[60] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[61] Thang Luong, Richard Socher, and Christopher Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 104–113, 2013.

[62] Larry R Medsker and LC Jain. Recurrent neural networks. 2001.

[63] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[64] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[65] Michael Mohler, Razvan Bunescu, and Rada Mihalcea. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 752–762. Association for Computational Linguistics, 2011.

[66] Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. Distinguishing antonyms and synonyms in a pattern-based neural network. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 76–85, 2017.

[67] Niels Ott, Ramon Ziai, Michael Hahn, and Detmar Meurers. Comet: Integrating different levels of linguistic modeling for meaning assessment. In *Proceedings of the 7th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2013, Atlanta, Georgia, USA, June 14-15, 2013*, pages 608–616, 2013.

[68] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, 2005.

[69] Pitambar Paudel. Online education: Benefits, challenges and strategies during and after covid-19 in higher education. *International Journal on Studies in Education*, 3(2):70–85, 2021.

[70] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014.

[71] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237, 2018.

[72] Jason Phang, Thibault Févry, and Samuel R Bowman. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

[73] Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online, July 2020. Association for Computational Linguistics.

[74] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[75] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, pages 337–346. ACM, 2011.

[76] Altaf Rahman and Vincent Ng. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789, 2012.

[77] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016.

[78] Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational*

Applications, BEA@EMNLP 2017, Copenhagen, Denmark, September 8, 2017, pages 159–168, 2017.

[79] Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pages 90–95, 2011.

[80] Alexander M Rush, SEAS Harvard, Sumit Chopra, and Jason Weston. A neural attention model for sentence summarization. In *ACLWeb. Proceedings of the 2015 conference on empirical methods in natural language processing*, 2017.

[81] Swarnadeep Saha, Tejas I. Dhamecha, Smit Marvaniya, Renuka Sindhgatta, and Bikram Sengupta. Sentence level or token level features for automatic short answer grading?: Use both. In *Artificial Intelligence in Education - 19th International Conference, AIED 2018, London, UK, June 27-30, 2018, Proceedings, Part I*, pages 503–517, 2018.

[82] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*, 2019.

[83] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[84] Mark D Shermis and Jill C Burstein. *Automated essay scoring: A cross-disciplinary perspective*. Routledge, 2003.

[85] André Smolentzov. Automated essay scoring: Scoring essays in swedish, 2013.

[86] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic

compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[87] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*, 2020.

[88] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2440–2448, 2015.

[89] Md. Arafat Sultan, Cristobal Salazar, and Tamara Sumner. Fast and easy short answer grading with high accuracy. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 1070–1075, 2016.

[90] Chi Sun, Luyao Huang, and Xipeng Qiu. Utilizing bert for aspect-based sentiment analysis via constructing auxiliary sentence. *arXiv preprint arXiv:1903.09588*, 2019.

[91] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[92] Kaveh Taghipour and Hwee Tou Ng. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Nat-*

ural Language Processing, pages 1882–1891, Austin, Texas, November 2016. Association for Computational Linguistics.

[93] Yi Tay, Minh C. Phan, Luu Anh Tuan, and Siu Cheung Hui. Skipflow: Incorporating neural coherence features for end-to-end automatic text scoring. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence*, pages 5948–5955, 2018.

[94] Wilson L Taylor. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433, 1953.

[95] Hao Tian, Can Gao, Xinyan Xiao, Hao Liu, Bolei He, Hua Wu, Haifeng Wang, and Feng Wu. SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4067–4076, Online, July 2020. Association for Computational Linguistics.

[96] Julien Tissier, Christophe Gravier, and Amaury Habrard. Dict2vec: Learning word embeddings using lexical dictionaries. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 254–263, 2017.

[97] Sophie Vanbelle. A new interpretation of the weighted kappa coefficients. *Psychometrika*, 81(2):399–410, 2016.

[98] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[99] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[100] Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4465–4476, 2019.

[101] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

[102] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Jiangnan Xia, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. *arXiv preprint arXiv:1908.04577*, 2019.

[103] Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. Structbert: Incorporating language structures into pre-training for deep language understanding. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[104] Yucheng Wang, Zhongyu Wei, Yaqian Zhou, and Xuanjing Huang. Automatic essay scoring incorporating rating schema via reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 791–797, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[105] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph and text jointly embedding. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1591–1601, 2014.

[106] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

[107] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[108] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, 2018.

[109] Craig Wishart and Retta Guy. Analyzing responses, moves, and roles in online discussions. *Interdisciplinary Journal of E-Learning and Learning Objects*, 5(1):129–144, 2009.

[110] Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. Rc-net: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1219–1228, 2014.

[111] Han Xu, Zhang Zhengyan, Ding Ning, Gu Yuxian, Liu Xiao, Huo Yuqi, Qiu Jiezhong, Zhang Liang, Han Wentao, Huang Minlie, et al. Pre-trained models: Past, present and future. *arXiv preprint arXiv:2106.07139*, 2021.

[112] Keyulu Xu, Mozhi Zhang, Jingling Li, Simon Shaolei Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. How neural networks extrapolate: From feedforward to graph neural networks. In *9th International Conference on*

Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021.

[113] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[114] Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[115] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 545–550, 2014.

[116] J Yuan and C Kim. Guidelines for facilitating the development of learning communities in online courses. *Journal of Computer Assisted Learning*, 30(3):220–232, 2014.

[117] Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. Swag: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, 2018.

[118] Xiangyang Zhou, Lu Li, Daxiang Dong, Yi Liu, Ying Chen, Wayne Xin Zhao, Dianhai Yu, and Hua Wu. Multi-turn response selection for chatbots with deep attention matching network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 1118–1127, 2018.