Learned in Translation: Contextualized Word Vectors

E4040.2018Fall.COWV.report
Xiaohui Li xl2694, Wenqi Jiang wj2285, Boyuan Sun bs3113
Columbia University

Abstract

We replicate the paper 'Learned in Translation: Contextualized Word Vectors': build attentional neural machine translation model, generate Context Vector (CoVe) and use CoVe as initialization on Natural Language Processing tasks. Though the amount work is huge while our computational resources are limited, we divide the work to individual and set a fixed amount of training steps. The result shows that given a fixed training step, CoVe outperforms GloVe as word vector initialization on both Question Answering and Classification tasks.

1. Introduction

Word vectors play an essential role in various Natural Language Processing tasks such as machine translation and classification tasks. Early researches [7][8] mainly focus on training word vectors by predicting contexts. Once the training process end, the word vectors would not be changed any more, which means the word vectors cannot adjust in different contexts. Thus, some researches [2][3] discuss contextualized word representation. Using these methods, the word vectors can benefit from referring to context words. Experiments show that Context Vector performs better than fixed word vectors such as GloVe [8].

In our project, we will rebuild the Context Vector [2] and feed it into other Natural Language Processing tasks such as classification and Question Answering. There are mainly two difficulties exist in our work. First, we need to build multiple models, including attentional neural machine translation model, bi-attentive classification network and dynamic co-attention network, which require huge amount of work. Second, our computational resources are limited. We have modest amount of GPU resources and time.

To solve these problems, we control our training steps to a fixed number and time. Applying the CoVe to specific tasks, we compare the results with GloVe initialization. Due to those methods, we avoid training the network for a large amount of time, usually weeks, and successfully generate comparable result for our tasks.

2. Summary of the Original Paper

In this section, we will first summarize the methodology provided by the original paper [2], including training neural machine translation model, generating Context Vectors and using CoVe on other tasks. Then, we will review the results on Classification and Question Answering tasks achieved by the authors.

2.1 Methodology of the Original Paper

Contextualized Word Vectors (CoVe) is one of the initializing methods for word vectors. Instead of using word as unit such as GloVe, CoVe takes a set of sequence as input, which provides more information than a single word input. The framework of our project is shown in Figure 2.1.

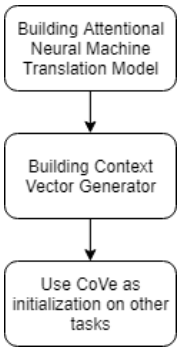

Figure 2.1 Simplified framework of the project

The first step of building CoVe is building neural machine translation model. Though attention mechanism is optional, an attentional sequence-to-sequence model is preferred [6]. The author feed GloVe word vectors as input to a 2-layer bidirectional LSTM. Then, the encoder-decoder structure would cooperate with attention mechanism and generate probability distribution of output words. The program then do beam search and generate a sequence of output. For dataset, the author used English-to-German corpus.

The second step is generating CoVe vectors. CoVe is basically the last layer output of encoder for each input word. Given a sequence of word, e.g. a sentence or a passage, the bidirectional LSTM encoder will generate a set of vectors on the top layer corresponding to each input word, which is defined as Context Vectors in the literature. The benefit of CoVe over other word vector initialization such as Word Vector [7] and GloVe [8] is that it benefits not only from the word meaning it self, but also the context information. In this case, CoVe can adjust word vectors based on the context, while other word vectors are fixed once training process finish.

The final step is using CoVe as initialization for NLP tasks such as Question Answering and Classification. The author used Bi-attentive Classification Network (BCN) on

Classification tasks and Dynamic Co-attention Network (DCN) on Question Answering tasks.

2.2 Key Results of the Original Paper

2.2.1 Classification and Question Answering

The author mainly compared 3 version of initialization for Question Answering and Classification tasks: random initialization, GloVe and GloVe+CoVe.

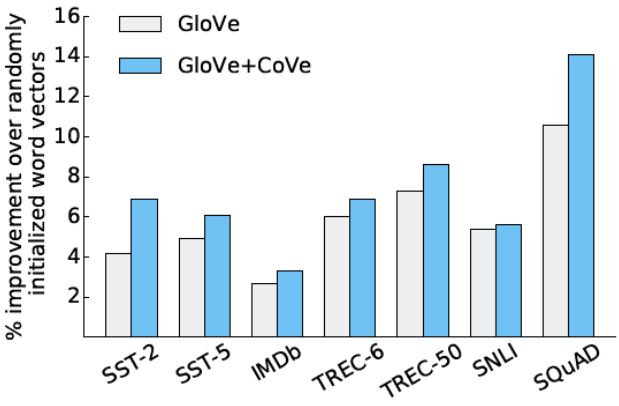

Figure 2.2 Performance improvement using CoVe and GloVe as initialization

As shown in Figure 2.2, both GloVe and GloVe+CoVe methods perform better than random initialization. Though GloVe is fairly better than random initialization, GloVe+CoVe performs even better in all the 7 tasks.

			GloVe+				
Dataset	Random	GloVe	Char	CoVe-S	CoVe-M	CoVe-L	Char+CoVe-L
SST-2	84.2	88.4	90.1	89.0	90.9	91.1	**91.2**
SST-5	48.6	53.5	52.2	54.0	54.7	54.5	**55.2**
IMDb	88.4	91.1	91.3	90.6	91.6	91.7	**92.1**
TREC-6	88.9	94.9	94.7	94.7	95.1	95.8	**95.8**
TREC-50	81.9	89.2	89.8	89.6	89.6	90.5	**91.2**
SNLI	82.3	87.7	87.7	87.3	87.5	87.9	**88.1**
SQuAD	65.4	76.0	78.1	76.5	77.1	79.5	**79.9**

Table 2.1 Comparing random initialization, GloVe and several versions of CoVe

In Table 2.1, the author further discusses about several versions of CoVe, which are trained by different size of datasets. Statistically, the CoVe generated by the largest dataset performs better than other CoVe and other initialization methods in most tasks. Also, when combining CoVe-L and character n-gram embeddings as initialization, the networks achieve the best result.

2.2.2 Machine Translation

The authors train their machine translation model on 3 datasets separately. They use WMT 2016 Multi30k dataset as the small dataset and achieve a BLEU score of 38.5. For medium dataset, they use IWSLT and achieve a BLEU score of 25.54. The large dataset and the corresponding BLEU score are WMT 2016 and 28.96. Even if the model performs best on the small dataset, the translation task of small dataset may be easier than other datasets.

3. Methodology

In this section, we will discuss about: how we implement machine translation; how to generate CoVe; and how we build neural networks for Question Answering and Classification tasks.

3.1. Objectives and Technical Challenges

In this project, we mainly focus on several objects:

1. Train an attentional sequence-to-sequence machine translation model on English-to-German datasets. In our project, we use OpenNMT [6] and choose the attention mechanism of Luong [4].
2. Given any English inputs, generate corresponding Context Vectors
3. Build bi-attentive classification network (BCN). Train the BCN on the Stanford Question Answering Dataset (SQuAD) using two kinds of initializations: CoVe and GloVe. Then compare their performance.
4. Build Dynamic Co-attention Network (DCN) [9]. Train the DCN on SST dataset. Compare the network performance under 2 different initializations, CoVe and GloVe.

This project is challenging mainly because it consists of so many parts: neural machine translation with attention mechanism, generating CoVe and feed it into other tasks, building bi-attentive classification networks and building dynamic co-attention network. Another challenge is the computational resources it takes: each neural network needs tremendous amount of time to train. For example, for the attentional neural machine translation model [6], the author use Titan V and train the model for 5 days, while we only have Tesla K80 with limited finance support.

Thus, we used two method to solve these problems. First, we split the work to each individuals first and construct them together later. Second, while we train the neural machine translation for 3 days, we spend much less time, e.g. 12 hours, on classification and question answering tasks. We set a max amount of step of training process and observe whether CoVe initialization performs better than GloVe.

3.2. Problem Formulation and Design

Below is the flowchart of our project. The first step of our project is to build bidirectional neural machine translation model with attention mechanism. Then, we will use the trained neural machine translation model to generate

CoVe corresponding to any given English inputs. Also, we need to build Dynamic Co-attention Network and Bi-attentive Classification Network for Question Answering and Classification tasks separately. Finally, we will compare CoVe and GloVe as initialization on SQuAD and SST dataset.

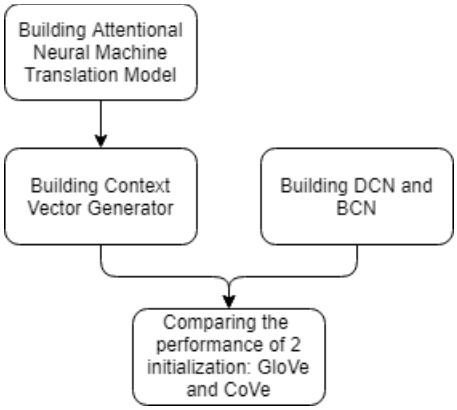

Figure 3.1 Project overview

Figure 3.2 shows the process of generating CoVe. We feed a sequence of English inputs to the trained NMT model and extract the last layer outputs of encoder as the corresponding CoVe.

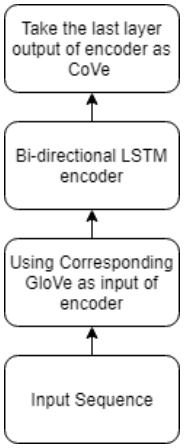

Figure 3.2 Generating Context Vector using trained Neural Machine Translation Model

4. Implementation

In this section, we will first discuss the network structures used in our project, including attentional neural machine translation model, bi-attention classification network and dynamic co-attention network. Then, we will provide our training details, including hyperparameters and other techniques. Finally, we will declare the datasets we use.

4.1. Deep Learning Network

4.1.1 Attentional neural machine translation model

Sequence-to-sequence machine translation model [10] achieved the attention of Natural Language Processing researchers. However, it may not be good enough dealing with long term dependencies, even if it uses LSTM. Thus, several attention-mechanism-based neural machine translation models has been proposed [4][5]. In this paper, we use the attention mechanism proposed by Luong et al. [4]. In this model, Luong improved the attention mechanism by using both global attention and local attention. As shown in Figure 4.1, global attention mechanism computes the probability distribution over all input words. In this case, the attention mechanism would compute the possibility at the place it should not look at, which negatively influence the model performance. Thus, [4] proposed the local attention mechanism. Figure 4.2 shows that local attention mechanism only focusses on a subset of inputs. The network first decides roughly where to look, then compute probability distribution within the window. This model was proved performing better than original attention mechanism [4][5].

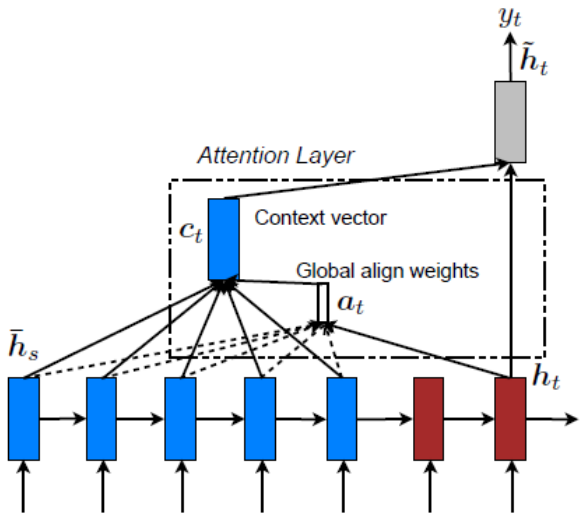

Figure 4.1 Global attention model

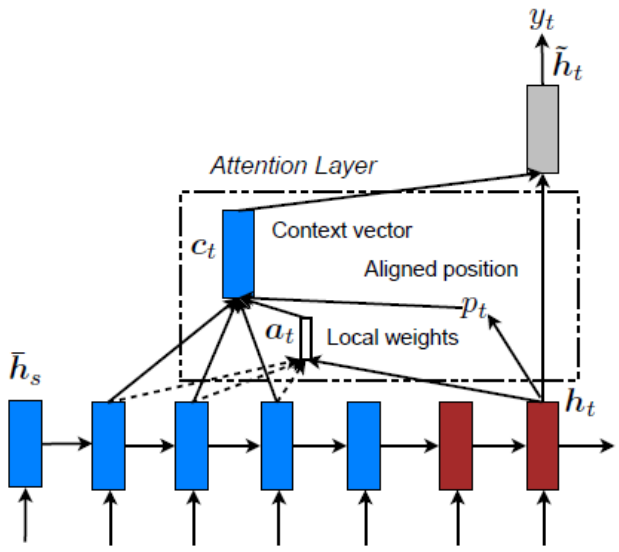

Figure 4.2 Local attention model

4.1.2 Global Vectors for Word Representation

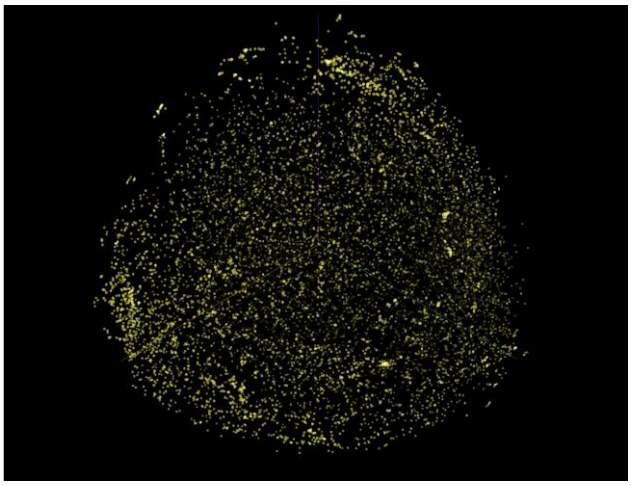

Figure 4.4 t-SNE 3D Visualization of the SQuAD dataset words mapping in the GloVe vectors

GloVe is a global log bilinear regression model that combines the advantages of the two major model families in the literature: global matrix factorization and local context window methods. This model benefits from both statistical information and context windows. Experiments show that GloVe outperforms other word embeddings such as [7] on similarity tasks and named entity recognition.

4.1.3 Context Vector

After building the neural machine translation model, we can generate Context Vector for any English input sequence. As shown in Figure 4.3, Context Vector is simply the last layer output of the NMT encoder: we feed input sequences to the encoder of trained NMT model, then grab the last layer outputs. The Context Vectors will then be feed into other tasks as word vector initialization.

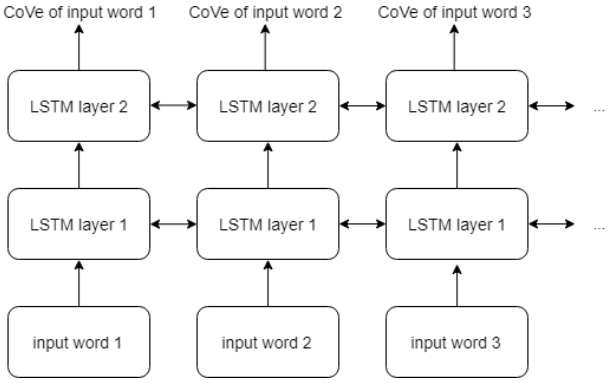

Figure 4.3 Context Vector

4.1.4 Bi-attentive classification network (BCN)

The BCN network is used for classification tasks. This sentence can handle one-sentence or two-sentences tasks. As the bottom of Figure 4.4, the network takes 2 sequences as inputs: if the input is one sentence, we replicate it as two. Then, the network implemented bidirectional attention mechanism, and used integrate as well as pooling layers. Finally, we use the maxout network to predict which class the input belongs to.

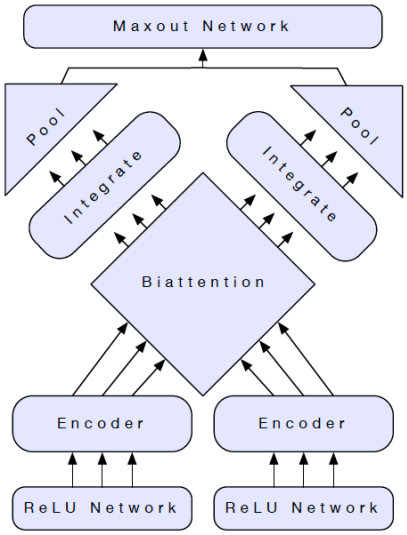

Figure 4.4 Bi-attentive classification network

4.1.5 Dynamic Coattention Network (DCN)

The DCN model is applied for question answering task. It divides the model into encoding and decoding component. For the encoder part, it designs a co-attention layer to capture the mutual dependence between document context and question and pass the compressed information within concatenated hidden layers into later decoder component. As for the decoder part, the model proposes a dynamic pointing decoding method, which incorporates an LSTM

layer to iteratively output the predicted starting and ending position and gives the final global optimum prediction.

Co-attention mechanism aims to determine the simultaneous interaction between the document and question per se. An affinity matrix fused from encoded document embedding as well as question embedding states is computed at each time step, and cointegrate with both the document and question to give the corresponding attention weight vectors, which through weighted sum derive the respective final attention-grained states. Attention states and input embedding states are lastly concatenated together as the compressed intermediate states.

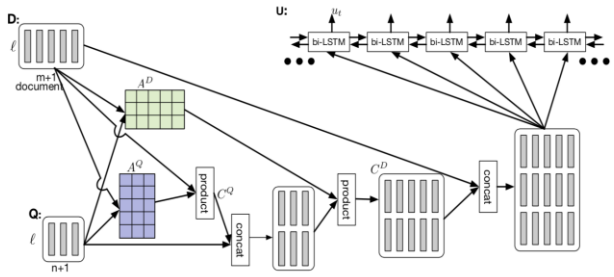

Figure 4.5 Co-attention Layer

Dynamic pointers decoder is designed to address the setback of local minima from simple generation of independent prediction of both answer starting and ending index. In general, it takes into account of the starting and encoding simultaneously and also their sequential dependence through a LSTM structure, moreover, a Highway Maxout Network with empirical proof is introduced as the final output layers.

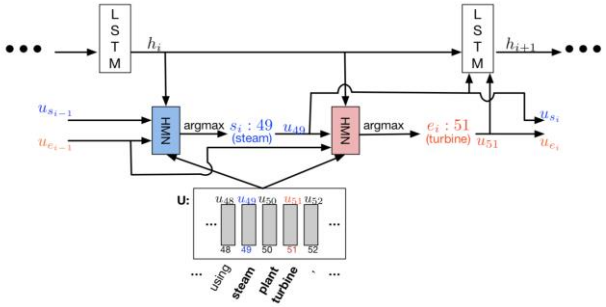

Figure 4.6 Dynamic Pointers Layer

4.2 Implementation details

4.2.1 Attentional Neural Machine Translation

There are several kinds of attention-based neural machine translation models [4][5]. The global attention model [5] is well but its performance may not be as good as [4]. [4] presents several kinds of attentional model. We choose local attention mechanism without scaling, so that the model performance could be promised while the trainable parameters would not be too much.

When it comes to hyperparameters tuning, we use the set of parameters provided by the original paper. We set the learning rate of 0.001, a dropout rate of 0.2, a batch size of 128 , and use the Adam Optimizer. The double-layer bidirectional LSTM contains 300 units in each hidden state.

4.2.2 Bi-attentive Classification Network

Bi-attentive classification network is designed to address the double context-based classification task with two text source and one classification output. In this research, we select the more representative Stanford Sentiment Treebank as our sentiment analysis classification task, which only has one sentence text source as input. Therefore, in the light of original work from paper, we regard the two same sentence as different input and input them into the double input, thus enabling a novel self-attention mechanism.

As illustrated from the paper, the CoVe encoding layer is implemented to replace the original shared double bidirectional LSTM layer for encoding the input GloVe embedding data. A comparison of performance for GloVe and GloVe+CoVe are mainly studied during real training, to see whether pre-trained CoVe layer is more efficient to generalize the pattern of input text embeddings.

4.2.3 Dynamic Coattention Network

We introduce dynamic coattention network (DCN) to address with the question answering task, which is specifically designed to relate the document and question text with co-attention mechanism.

Apart from implement the original DCN model, we build up the pretrained CoVe encoding layer between GloVe embedding layer and the co-attention layer to reinforce the encoding ability.

4.3 General Deep Learning Framework

In face of the multiple-tasks with large-scaling data, we design a general framework to facilitate the programming implementation of network models. In high level, we divide the deep learning implementation task into data process, data feed, network architecture, model training and interaction five component. Specifically, we used Objected Oriented Programming modules to implement the abstraction of the five part.

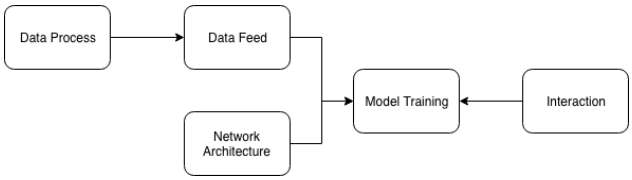

Figure 4.7 General implementation framework

As for the data process and feed part, we wrote programs to automatically download the dataset data from internet and conduct general text process, transforming raw text data into tokenized index from embedding word vector. GloVe vector is meanwhile incorporated saving as a embedding matrix. In terms of data feed procedure, a data pipeline object is created to save the data from local file into memory and could repeatedly and randomly input task-tailored batch data for training procedure.

Network architecture represents the implementation details of different models, containing the structure at the layer level. For instance, attention mechanism layer is built up at the architecture module with detailed functionality. It is largely built up from tensorflow fundamental functions and mainly complements the work of model training.

Model training systematically enables the general training procedure of deep learning task, containing the building of graph, loss defining and optimization operation. It reflects the tensorflow training procedure.

Interaction aims to make the training and evaluation step more efficient from three fields: hyper-parameterization, model functionality and training monitor. Concretely, tensorflow built-in *tf.app.flags* system is utilized to define and control all the structure and training parameter features, meanwhile model training, evaluation and result demonstration is realized with the terminal program as python file. Training monitor is shown on tensorboard at server to derive real-time information with easy online access.

4.4 Datasets

4.4.1 Attentional Neural Machine Translation

We use two datasets of different scales: the small dataset WMT16 Multi30K and the large dataset WMT16.

4.4.2 Classification

We used Stanford Sentiment Treebank dataset (SST) [12] to study model performance in classification. SST the corpus of movie review excerpts from the *rottentomatoes.com* website originally collected and published by Pang and Lee [13].

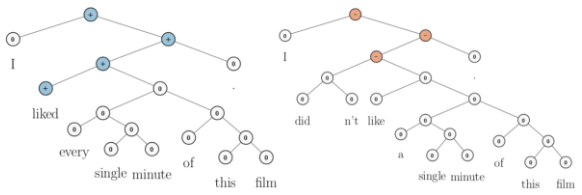

Figure 4.8 Examples of Tree Sentiment Data

The original data was in the form of tree structure with phrase-sentiment pair and we preprocess the dataset into derive the sentence-sentiment level data as a standard sentiment classification task. In total we process 10k observations and split according to the 8:1:1 ratio into training, validation and development set.

The dataset finally includes 10,662 sentences with continuous sentiment scores ranging from 0 to 1. We further discretize the sentiment into classification types with complex five positions: negative, somewhat negative, neutral, somewhat positive or positive; (SST-5) and simple two positions: negative and positive. (SST-2) Note that for SST-2 we specifically eliminate the neutral (sentiment scores at [0.4, 0.6)) labeled data.

4.4.3 Question Answering

We studied our question answering model with the Stanford Question Answering Dataset (SQuAD) [14]. Specifically, for compatibility with the original results, we used the previous 1.1 version instead of the newly updated 2.0 version [15].

SQuAD contains two input text: the document text and the question text, with a answering part which is inside the document. The document is at paragraph length with the word length of 200-400, the question is a short sentence, while the answer is a short phrase with several words. The dataset requires the model to be able to understand the logical relationship between content of document and question to give out the right answer (starting and ending position index).

Figure 4.9 Examples of SQuAD Data

In terms of the model evaluation we used F1 Scores (Macro-average), which the average overlap between the prediction and ground truth answer. We treat the prediction and ground truth as bags of tokens and compute their F1. And EM Scores (Exact match) which measures the percentage of predictions that match any one of the ground truth answers exactly.

5. Results

In this section, we will discuss our results of neural machine translation, Classification and Question Answering tasks. Then, a result comparison with original paper will be presented. Finally, we will discuss our result on several aspects.

5.1. Project Results

5.1.1 Neural machine translation

We train our attentional neural machine translation model on both large (WMT16) and small (WMT16 Multi30K) datasets. After 2-day training, both of these models achieved best BLEU scores over 4 on the same test set. We present the result of both training dataset below, as shown in Figure 5.1 to Figure 5.4.

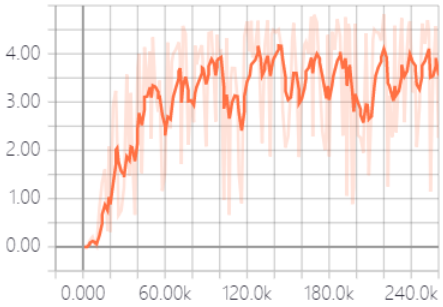

Figure 5.1 BLEU score on test set of our NMT model trained by small dataset

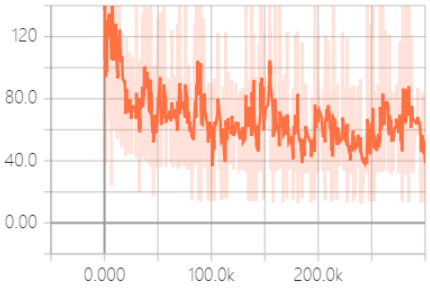

Figure 5.2 Loss value over time of our NMT model trained by small dataset

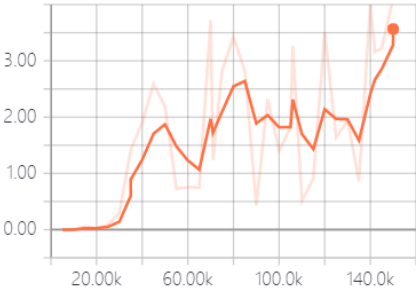

Figure 5.3 BLEU score on test set of our NMT model trained by large dataset

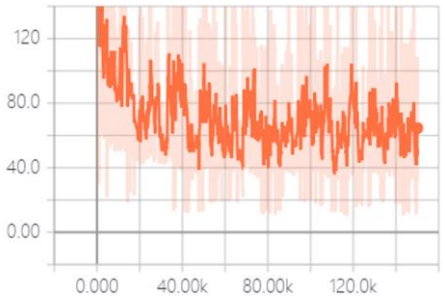

Figure 5.4 Loss value over time of our NMT model trained by large dataset

As we can see in Figure 5.1 and Figure 5.3, the loss value descends from over 120 to around 60 after 2-day training. The BLEU score increases from 0 to around 4 as shown in Figure 5.2 and Figure 5.4.

Notice that even if we train both model for 48 hours, the model experience around 240k steps using small dataset while only about 150k steps on large dataset. This is attribute to the sentence length of different datasets.

5.1.2 Classification

We train our model on both the processed simple SST-2 and complex SST-5 dataset with respective GloVe-BCN and GloVe-CoVe-BCN model.

The model in practice suffers enormously from the overfitting issue. Even though with hard tuning of dropout probability and less hidden state size, the model still underperforms at the development set for evaluation.

The performance metrics of our trained models is at best accuracy of 53% for the SST-2 and accuracy score of 24% for the SST-5. The loss learning convergence is still at difficulty after several improvements of larger training batch-size, learning rate decay.

Figure 5.5 The longest training time loss value of our BCN model on the development dataset.

The problem could be mainly attributed to the limited size of dataset, which only has 7k training short sentence text sequence. The capacity of the model generalization availability is limited, and our complex models do poorly. To specify the training details, we fine tune the learning rate to be 0.01, 0.005 and 0.001, dropout rate for each layer starting from 0.1 to 0.6 and the hidden state size

from 150 to 100 and lastly decrease to 50, far less than the CoVe layer dimension of 300.

However, after controlling for the same parameters, the relative result improvement of GloVe+CoVe over GloVe only is still clear in that the CoVe vector is superior in improving the classification, even though the model itself fails to learn well the small text dataset. We present the respective SST-2 and SST-5 development evaluation results as below:

Figure 5.6 Accuracy on development set of our GloVe (pink line) and GloVe+CoVe model (green line) at SST-2 dataset.

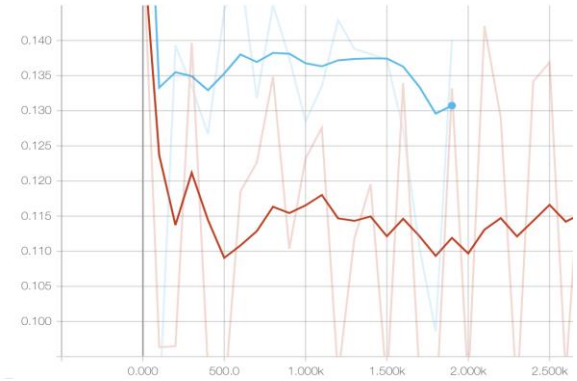

Figure 5.7 F1 scores on development set of our GloVe (red line) and GloVe+CoVe model (blue line) at SST-5 dataset.

5.1.3 Question Answering

The SQuAd dataset has much larger scale and more computation costing. We therefore divide our study into two components: short-time control-group experiments to test the performance of CoVe and long-term single training to derive the best model.

Given same set of hyperparameters as well as training environment, we respectively check the training situation of GloVe and GloVe-CoVe. Under empirical practice, the training time of each step for CoVe based model is significant longer than the None-Cove, yet CoVe converges faster under Adam optimizer and the development set metrics of both SQuAd F1 and EM are comparatively better.

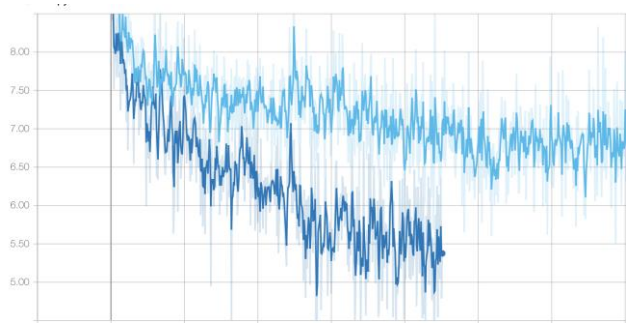

Figure 5.8 Development set loss on our GloVe (light blue) and GloVe+CoVe model (dark blue) at SQuAD dataset given same training time

After validation of the CoVe improvement on the DCN model, we search for best hyperparameters and implement CoVe model to find the best prediction model. The best model reaches the F1 score of 0.51 with EM of 0.48.

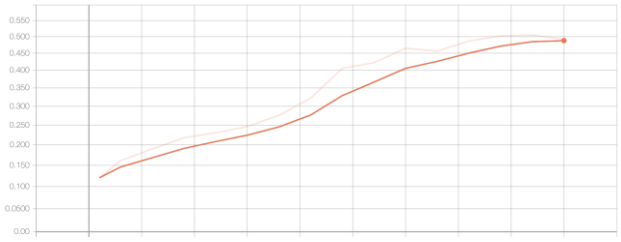

Figure 5.9 F1 scores over training time of our best GloVe-CoVe-DCN model at the development set.

5.2. Comparison of Results

5.2.1 Neural machine translation

After 2-day training on Tesla K80, our model achieves BLEU scores around 4 on both small and large dataset, while the original paper achieves BLEU score over 25. Though the authors do not mention their training time, we refer to the original attentional neural machine translation model [4] and find they use TITAN V and train the model for 5 days. That partially explains the large performance gap between our model and the author's.

5.2.2 Classification

Our implemented classification model is undermined by the limitedness of SST data scale and therefore perform less than the original paper results in terms of both SST-2 and SST-5 tasks. Given that we strive to fine-tuned with the hyperparameters, the large gap may mainly attributed to the over complexity of GloVe-CoVe-BCN model. Nevertheless, the same advantage of CoVe over Non-CoVe is validated at the development data set.

5.2.3 Question Answering:

The same edge of CoVe is also confirmed at our control-group experiments. And we utilized CoVe together with

DCN structure to derive a final 50% F1 score of best models. There is still difference between our tuned model and the original one (75%) and those superior in the SQuAd leader board (85%) [12]. The difference could be attributed to our scarceness of computation resource, and with more available time and finer hyper-parameterization there is potential to gain the best model.

5.3. Discussion of Insights Gained

5.3.1 Neural machine translation

We assume the low BLEU score of our attentional neural machine translation is due to the lack of performance of Tesla K80.

We trained our model on two datasets of different scale. Considering the converge speed of the attentional neural machine translation model, we first use a small dataset, WMT16 Multi30K. However, the best BLEU score stuck at 4.88 after 2-day training. Though this may attribute to the weak performance of Tesla K80, we decide to train the model on a larger dataset to see if there is any difference. Unfortunately, the training speed is almost same: in the first 12 hours, it get a best BLEU score of 3, but after that the improvement slow down and the best BLEU score end up at around 4.8 after 48-hour training. Thus, the scale of dataset may not be the main reason of the slow training speed in our project.

We could have tried different learning speed to see if the training speed can make a difference. However, due to the limited amount of time, we should move on to the CoVe part.

5.3.2 Classification

A classical predicament of overfitting is live presented at our GloVe-CoVe-BCN model, whose capacity suffers with the limitedness of dataset scale.

At length, the advantage of CoVe is still confirmed with comparative better performance, this concretely supports the notion that CoVe deep pre-trained layer is better than the original trainable Bidirectional LSTM layer into encoding the input GloVe embedding words.

5.3.3 Question Answering

The CoVe generalization ability is also validated, and we achieved a standard level performance, which shows the powerful capacity of our GloVe-CoVe-DCN model. However, there is still fair place for the improvement potential with respect to detailed hyper-parameters fine tuning, ensemble mechanism, etc.

From the empirical side, through the real experience it is more instructive that deep learning is also an art, we tune and improve details during training to make our model learn better, training methods as gradient clipping, decaying learning rate, larger batch size with smaller learning rate are empirically proved to be able to marginally improve the model.

6. Conclusion

In this project, we successfully explored the world of natural language processing by implement various NLP models in multiple tasks. We achieved building attentional neural machine translation model and generate CoVe for English corpus. We also built Dynamic Co-attention Network and Bi-attentive Classification Network for Question Answering and Classification tasks separately. Due to the limited computational resources, we only compared the converging speed of GloVe and CoVe initialization within a fixed amount of time. Results show that CoVe initialization still outperforms GloVe within a fixed amount of time, while using CoVe is a more time-consuming option. If we had more GPU resources, we could furthermore explore the converged network performance using different initialization methods.

6. Acknowledgement

We sincerely appreciate Professor Zoran Kostic, who gave us valuable suggestions on project planning, paper replicating and report writing.

7. References

[1] Bitbucket:https://jwQAQ@bitbucket.org/4750_zffp/project4040nndl.git

[2] McCann, Bryan, et al. "Learned in translation: Contextualized word vectors." *Advances in Neural Information Processing Systems*. 2017.

[3] Peters, Matthew E., et al. "Deep contextualized word representations." *arXiv preprint arXiv:1802.05365* (2018).

[4] Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." *arXiv preprint arXiv:1508.04025* (2015).

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

[6] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-source toolkit for neural machine translation. ArXiv e-prints, 2017.

[7] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[8] Pennington, Jeffrey, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation." *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014.

[9] Wieting, John, et al. "Towards universal paraphrastic sentence embeddings." *arXiv preprint arXiv:1511.08198* (2015).

[10] Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." *Advances in neural information processing systems*. 2014.

[11] C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. In

Proceedings of The 33rd International Conference on Machine Learning, pages 2397–2406, 2016.

[12] Socher, Richard, et al. "Recursive deep models for semantic compositionality over a sentiment treebank." *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013.

[13] Pang, Bo, and Lillian Lee. "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales." *Proceedings of the 43rd annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2005.

[14] Rajpurkar, Pranav, et al. "Squad: 100,000+ questions for machine comprehension of text." *arXiv preprint arXiv:1606.05250* (2016).

[15] Rajpurkar, Pranav, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD." *arXiv preprint arXiv:1806.03822* (2018).

[16] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805* (2018).

[]

8. Appendix

8.1 Individual student contributions in fractions - table

	xl2694	wj2285	bs3113
Last Name	Li	Jiang	Sun
Fraction of (useful) total contribution	1/3	1/3	1/3
Builing CoVe & NMT model	1/3	1/3	1/3
Building DCN & BCN	1/3	1/3	1/3
Report and proposal	1/3	1/3	1/3