Predicting Intensity of Emotions in Tweets with BERT

CSCI5922 Spring 2020 Neural NEtworks and Deep Learning

Xuan Gao, Ji Zhao

1. Introduction

Translating texts is very challenging for a machine since the text has the complicated words structures and different word sequences express completely different emotions.. Moreover, it becomes more difficult if the texts include irregular word orders, emojis, website url, and buzzwords. Social media contents provide abundant text sources for natural language processing (NLP) research, however, they have all these features.

Social media allows people to share events, opinions, and emotions efficiently. Twitter, as one of the most popular social media platforms, has more than 500 million tweets per day. These resources are valuable for Natural Language Processing due to the wide availability and novelties.

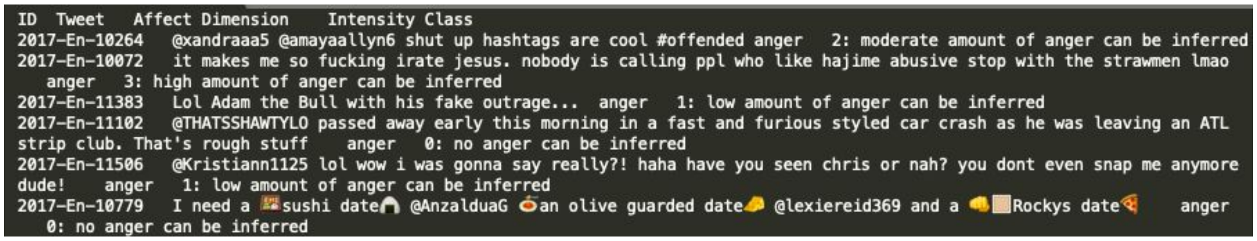
In this project, we used the labels data from SemEval-2018 Task 1: Affect in Tweets.[1] The features of the data include ID, Tweet, Affect Dimension, and Intensity Class. Among the four different languages, we chosed English Tweets to analyze.

2. Building on Past Work

The big amount of prior work on sentiment and emotion analysis (Mohammad, 2016) was used as reference in our project. For this task, teams that participated also provided many useful methods and instructions.([2][3][4]) However, there is no team using BERT in this task. Thus, we would like to use BERT in this task.

3. Overview

3.1 Data



```
ID  Tweet   Affect Dimension    Intensity Class
2017-En-10264   @xandraaa5 @amayaallyn6 shut up hashtags are cool #offended anger   2: moderate amount of anger can be inferred
2017-En-10072   it makes me so fucking irate jesus. nobody is calling ppl who like hajime abusive stop with the strawmen lmao
   anger   3: high amount of anger can be inferred
2017-En-11383   Lol Adam the Bull with his fake outrage...  anger   1: low amount of anger can be inferred
2017-En-11102   @THATSSHAWTYLO passed away early this morning in a fast and furious styled car crash as he was leaving an ATL
strip club. That's rough stuff    anger   0: no anger can be inferred
2017-En-11506   @Kristiann1125 lol wow i was gonna say really?! haha have you seen chris or nah? you dont even snap me anymore
dude!    anger   1: low amount of anger can be inferred
2017-En-10779   I need a 🍱sushi date🍙 @AnzalduaG 🍝an olive guarded date🧀 @lexiereid369 and a 👊🏼Rockys date🍕    anger
   0: no anger can be inferred
```

Figure 1-1. Training Data

The dataset is about emotion ‘anger’ with different intensity. The training dataset consists of 1701 entries and development dataset consists of 389 entries. As shown in Figure 1-1, there are three features, Tweet, Affection Dimension, and Intensity Class. We mainly used Tweet as our training text to predict the target - Intensity Class.

3.2 Preprocessing

The greatest convenience of social media is that users can share anything at anytime and anywhere. Thus, formal english is rarely used in their posts. Instead, they might use the wrong vocabulary and sentence's grammar. In addition, some irregular words, like emojis, hashtag, abbreviation words and web url make a sentence more complicated and hard to understand. After reviewing the data, as shown in Figure 1, we found these existed the above mentioned problems, so we decided to preprocess the data first.

3.2.1 Tokenize

We utilized the *ekphrasis* [1] (Baziotis et al., 2017) tool to process the Tweet part of the data. Ekphrasis can recognize tweets markup, dates and currencies. The processing steps included tokenization, word normalization, word segmentation (for splitting hashtags) and spell correction.

Here is an example from the *ekphrasis* website. The original tweet is:

```
I need a 🍱sushi date🍙 @AnzalduaG 🍝an olive guarded date🧀 @lexiereid369 and
a 👊🏽Rockys date🍕
```

After *ekphrasis* processing, the result is:

```
i need a 🍱 sushi date 🍙 <user> 🍝 an olive guarded date 🧀 <user> and a 👊 🏽
rockys date 🍕
```

Since tweets data that we used have emoji images but *ekphrasis* does not process them, we utilized *demoji* [2] package to convert emoji into words, the result is:

```
i need a : bento_box : sushi date : rice_ball : <user> : spaghetti : an olive
guarded date : cheese_wedge : <user> and a : oncoming_fist_medium -
light_skin_tone : rockys date : pizza :
```

Then we used tokenization of BERT to convert the sentence into word list, the final result is:

```
['i', 'need', 'a', ':', 'bent', '##o', '_', 'box', ':', 'su', '##shi', 'date',
':', 'rice', '_', 'ball', ':', '<', 'user', '>', ':', 'spaghetti', ':', 'an',
'olive', 'guarded', 'date', ':', 'cheese', '_', 'wedge', ':', '<', 'user', '>',
'and', 'a', ':', 'on', '##coming', '_', 'fist', '_', 'medium', '-', 'light',
'_', 'skin', '_', 'tone', ':', 'rocky', '##s', 'date', ':', 'pizza', ':']
```

3.2.2 Embedding

Word embedding are word vectors that can be used as features in natural language processing [5]

1. https://github.com/cbaziotis/ekphrasis
2. demoji

(Mikolov et al., 2013). We used BERT file `Run_classifier.py` [1] to convert the tokens into word embeddings. The result word vector is 3-dimension, which includes token ids, mask ids, and segment ids.
We used BERT base mode `bert_uncased_L-12_H-768_A-12` [2] (L=12, H=768, A=12, Total Parameters=110M) to convert words into their corresponding ids. Then we set a maximum length of sequence. Max sequence length is the maximum total input sequence length after WordPiece tokenization. Sequences longer than this number will be truncated, and sequences shorter than this will be padded as zero. [1] Since we did not consider the relationship between sentences, segment ids were not used in our project.

3.3 BERT

Traditional Recurrent Neural Network (RNN) computes along with symbols positions of the input and output sequence. The next computation needs to wait for the previous results, which inherently precludes parallely computation. Bert, which stands for Bidirectional Encoder Representations from Transformers, relies on self-attention to compute representation of input and output without using sequence-aligned RNNs. [8] Since there is no team used BERT to process the data, we want to use it to obtain the performance of BERT.

4. Model Description

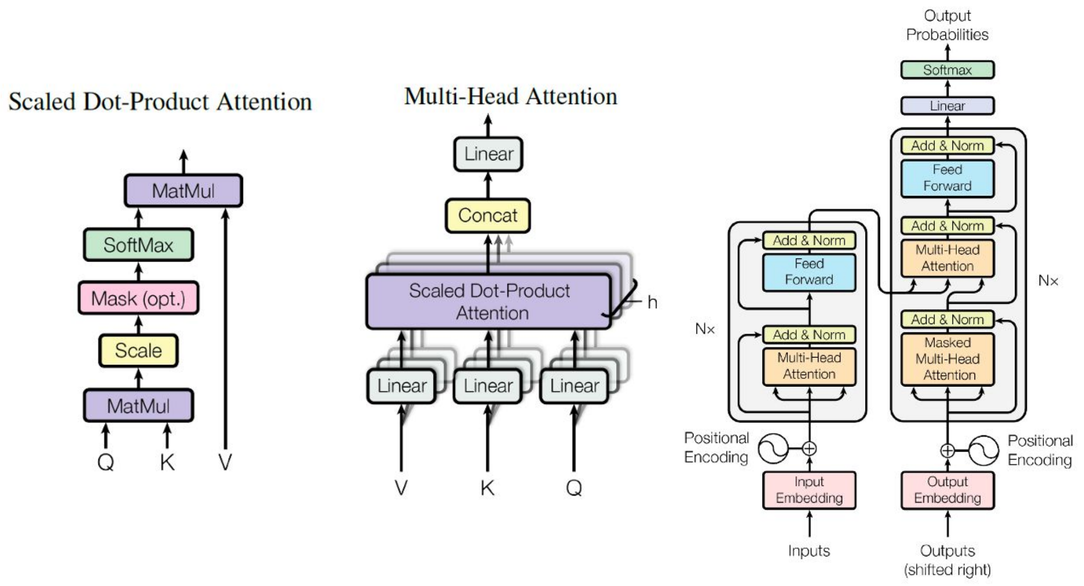


Figure 2-1. (left) Scaled Dot-Product Attention. (middle) Multi-Head Attention consists of several attention layers running in parallel. (right) The Transformer - model architecture.

1. bert/run_classifier.py at master · google-research/bert · GitHub
2. TensorFlow Hub

Bert is a derivative from the Transformer model which is a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output.

In general, Attention mappings can be described as a function of a query and a set of key-value pairs. Transformers use a ”Scaled Dot-Product Attention” to obtain the context vector. The Transformer also uses multi-head attention which allows the model to jointly attend to information from different representation subspaces at different positions.
BERT is a multi-layer bidirectional Transformer encoder. There are two steps in BERT: pre-training and fine-tuning. For pre-training, the model is trained on unlabeled data over different pre-training tasks. For fine-tuning, all initialized parameters are fine-tuned using labeled data from the downstream tasks. In our project, we used fine-tuning.

5. Experiments and Results

For the dataset that we used, we tuned the parameters max sequence length, epochs, and batch size. We also applied the classifier into the binary classification project and multi-classification project. Learning rate we used is 1e-5.

5.1 Multinomial Classification

5.1.1 Max Sequence Length

We set max sequence length at 8, 32, and 64. Then we tested it at different batch sizes and epochs. From Figure 5-1, we can obtain that when max sequence length is a constant, epoch equals 5 always has higher accuracy than epoch equals 10, and batch size equals 50 always has the best performance.

The higher epoch gives a lower accuracy, it may be due to that epoch equals 10 causes overfitting problem, which results in a lower accuracy. The exception happens at max sequence equals 8. The reason is that if max sequence length is too small, the more information will be lost. Thus, the accuracy is low and the results are uncertain.

5.1.2 Epoch

From Figure 5-2, we can obtain that when epoch is a constant value, the performance of the classifier increases when max sequence length increases, and batch size equals 50 always has the best performance.
If max sequence length is large, then word vectors could include more information. Thus, it will give a higher accuracy. In this case, we also tried max sequence length equals 128. However, we cannot perform it due to the limitation of disc storage on google drive.

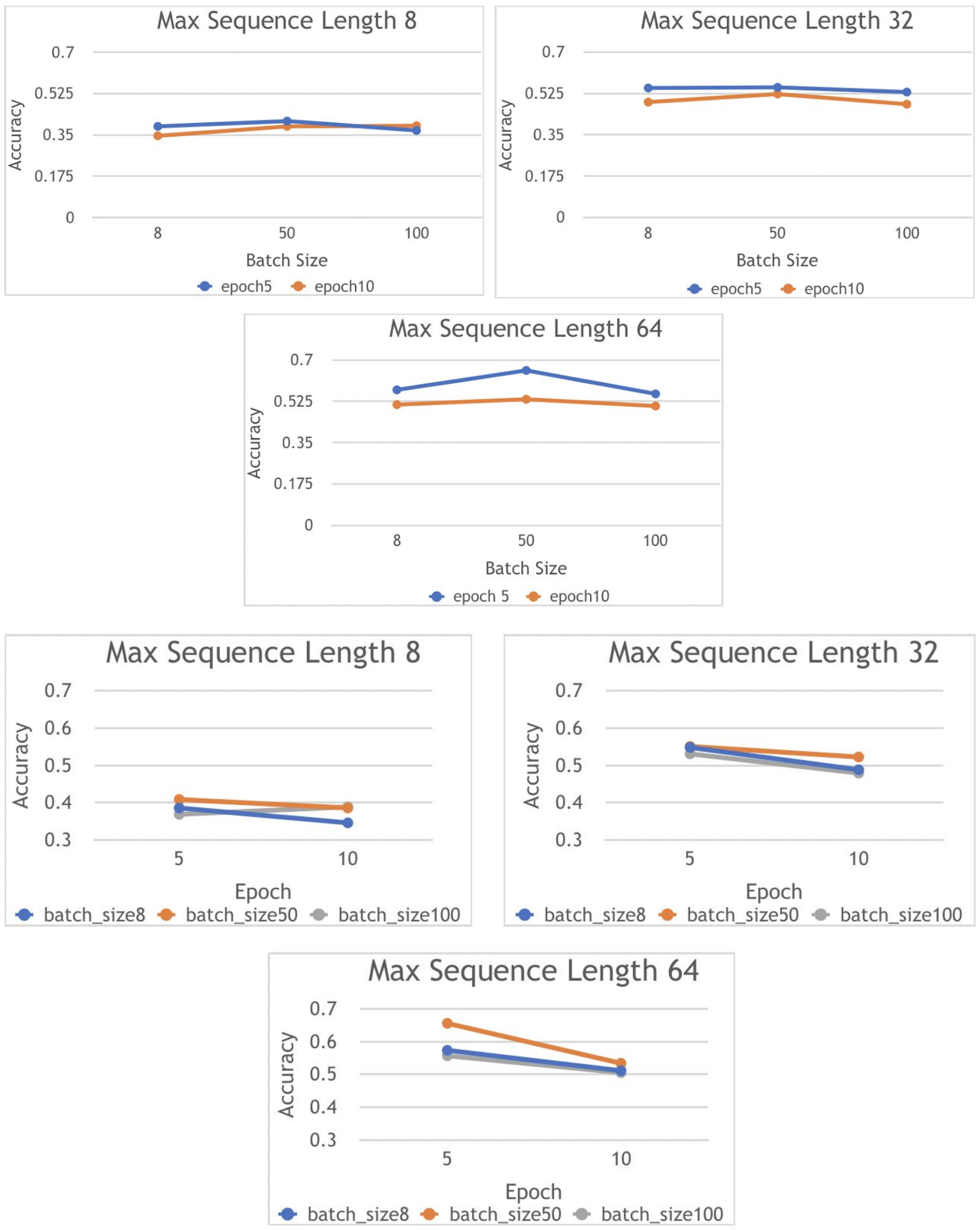


Figure 5-1 Accuracy comparison when max sequence length is a constant (multinomial classification) (top row) x axis is batch size (bottom row) x axis is epoch

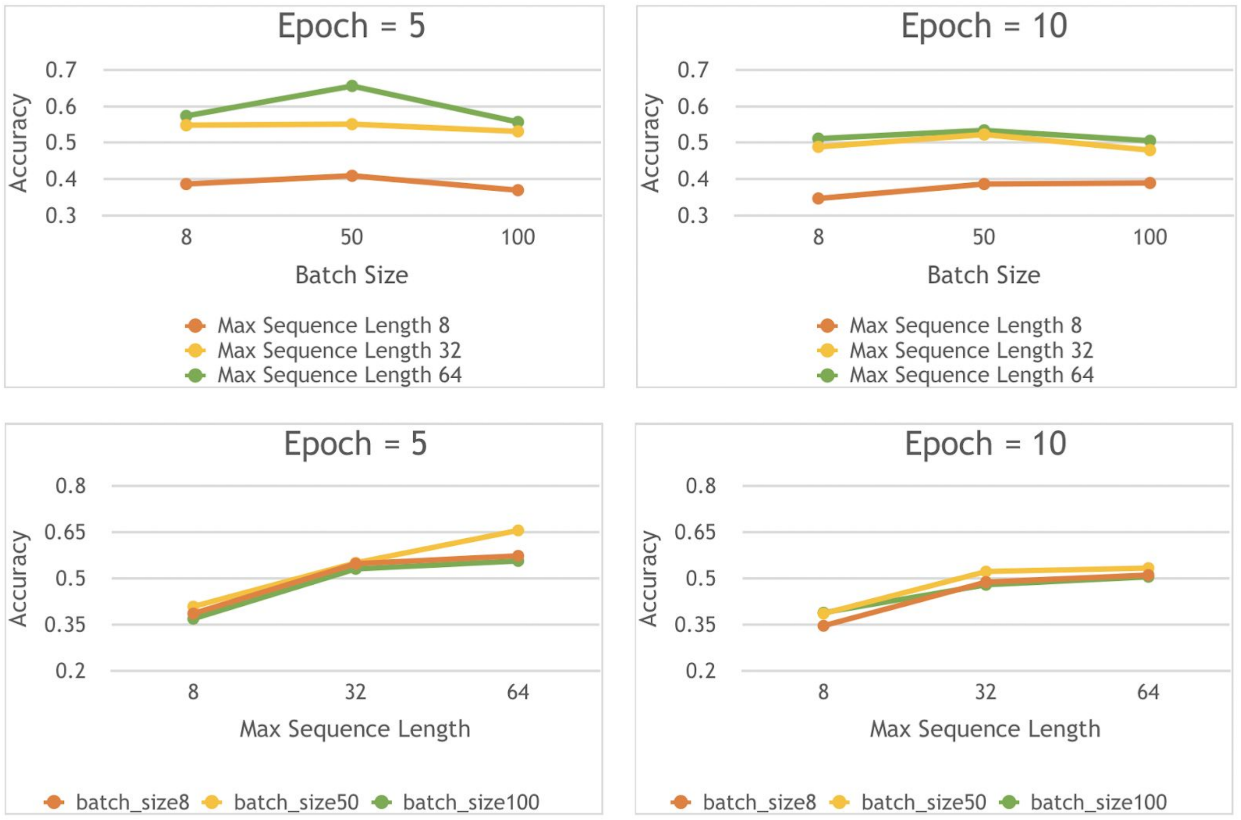
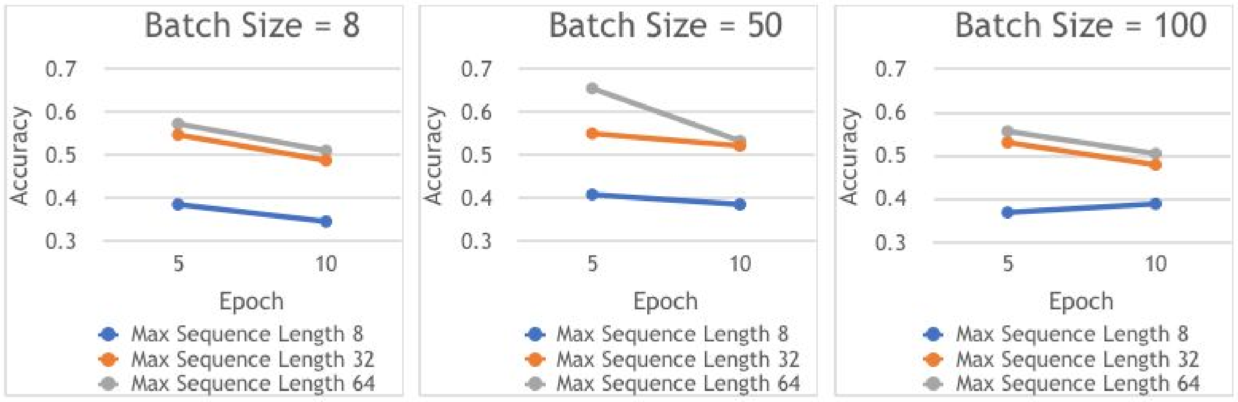


Figure 5-2 Accuracy comparison when epoch is a constant (multinomial classification) (top row) x axis is batch size (bottom row) x axis is max sequence length

5.1.3 Batch Size

From the above conclusions and Figure 5-3, it is easy to tell when batch size is constant, the performance of the classifier decreases with epoch increasing, and larger sequence length always gives the best result.



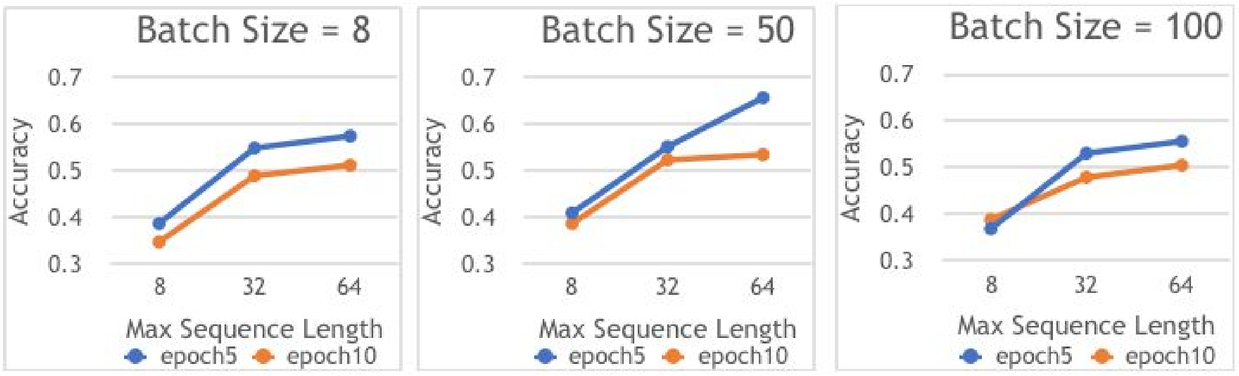


Figure 5-3 Accuracy comparison when batch size is a constant (multinomial classification) (top row) x axis is epoch (bottom row) x axis is max sequence length

5.2 Binary classification

For binary classification, we can obtain that when epoch is a constant, larger max sequence number gives a better result. For batch size, when it is greater than 50, the result does not change a lot no matter whether epoch equals 5 or epoch equals 10.

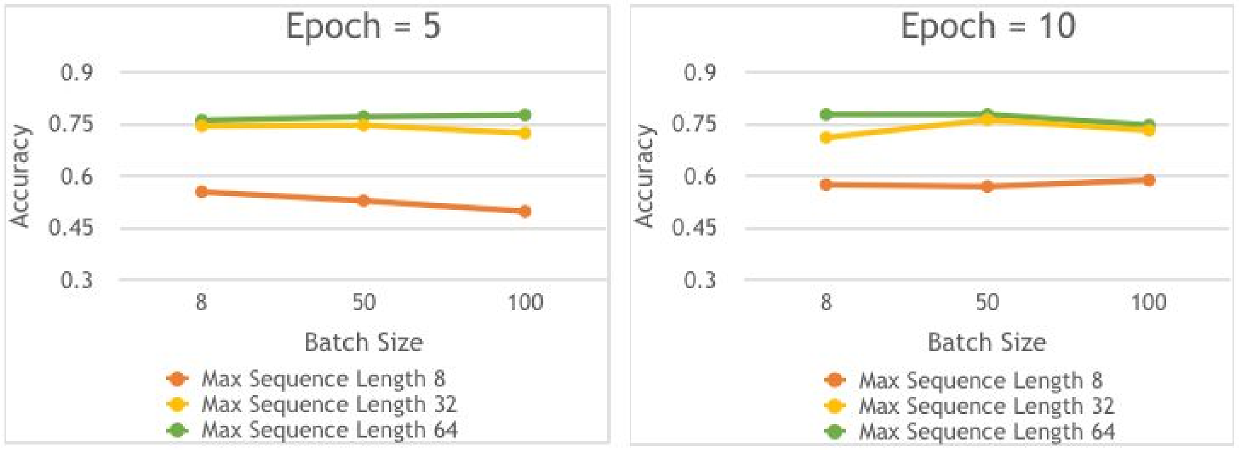


Figure 5-4 Accuracy comparison when batch size is a constant

5.3 Binary vs Multinomial Classification

From Figure 5-5, we can clearly obtain that the performance of binary classification is better than multinomial classification. Since the data size is small, the word vector cannot provide enough information to train a multinomial classifier. With the same number of entries, the binary classifier can obtain more information than multi-classification, which results in a higher accuracy.

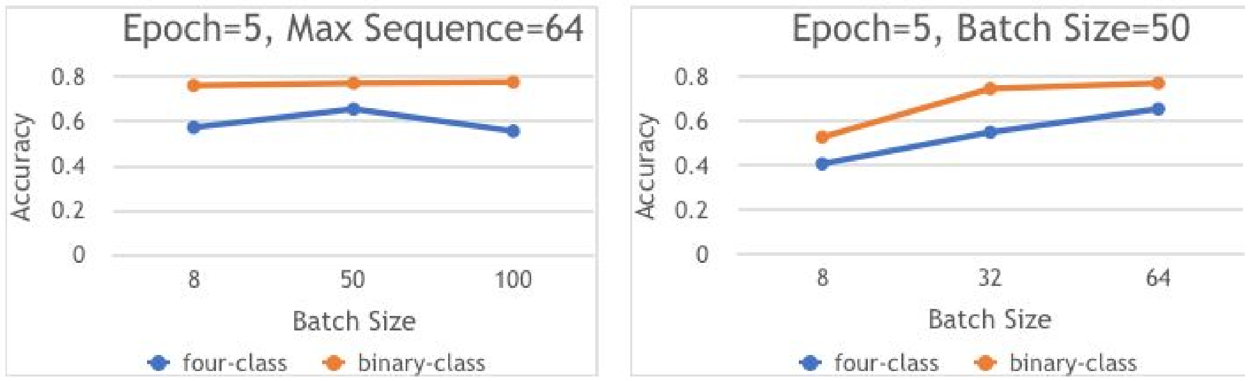


Figure 5-5 Accuracy comparison between multinomial and binary classification

6. Conclusion & Future works

From the experiments we did, we found that accuracy will increase if word vectors include more information, that is, if we don't consider the influences from the calculation efficiency and cost, the larger max sequence length can give the higher accurate result and better performance. In addition, an appropriate epoch and batch size can also help to find better accuracy.

In our project, we finished the task of multi-classification. However, the accuracy is about 50%, which is lower than the previous research results. Increased train steps in a proper range could improve the classifier's performance, but it may cause an overfitting problem when the train steps are too large.

Due to the limitation of our data and hardware configuration, the accuracy of multinomial classification is not high.We plan to collect more data in training the multinomial classifier.

In addition, most of our efforts were in figuring out the conflicts of the BERT model between TensorFlow1 and TensorFlow2, and finally, we implemented TensorFlow 1.15 in this project. For the next step, we will continue to fix the conflicts and try to implement it on TensorFlow 2 later.

7. References

[1] CodaLan Competition
[2] Baziotis, C.; Athanasiou, N.; Chronopoulou, A.; Kolovou, A.; Paraskevopoulos, G.; Ellinas N.; Narayanan, S.; Potamianos, A. NTUA-SLP at SemEval-2018 Task 1: Predicting Affective Content in Tweets with Deep Attentive RNNs and Transfer Learning. Proceedings of the 12th International Workshop on Semantic Evaluation, 2018.

[3] Park, J.H.; Xu, P.; Fung, P. PlusEmo2Vec at SemEval-2018 Task 1: Exploiting Emotion Knowledge from Emoji and #Hashtags. Proceedings of the 12th International Workshop on Semantic Evaluation, 2018.
[4] Duppada V.; Jain R.; Hiray, S. SeerNet at SemEval-2018 Task 1: Domain Adaptation for Affect in Tweets. Proceedings of the 12th International Workshop on Semantic Evaluation, 2018.
[5] Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. International Conference on Learning Representations, 2013.
[6]Vaswani A.; Shazeer N.; Parmar N.; Uszkoreit J.; Jones L.; Gomez A.; Kaiser L.; Polosukhin L. Attention Is All You Need. NeurIPS 2017.
[7] Devlin J.; Chang M.; Lee K.; Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019.