**Advanced Natural Language Processing**

**Minh Hieu Tran – Candidate number: 229560**

Data Science MSc student

School of Mathematical and Physical Sciences

Department of Mathematics

University of Sussex

1. **INTRODUCTION**

Microsoft sentence completion challenge, even though introduced in back in 2011, has been a very challenging task. In this challenge, one is given a question, which is sampled from five of Conan Doyle’s Sherlock Holmes novels, together with 5 potential answers in which 4 of them are impostor answers. In their paper, the highest score (accuracy) achieved is 49% with Latent Semantic Analysis Similarity model, whereas other simpler models like Trigram or Quadro-gram can only get to 34-39%. Performance by human in this dataset is only 91%, indicating that this is a difficult task, even with human evaluators.

In this paper, I will apply state-of-the-art language model currently, namely BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (Robustly optimized BERT approach); as well as a multilayers BiLSTM (Bidirectional Long short-term memory) to the problem in order to compare their performance to the baseline unigram/bigram model. Analysis of error will also be conducted, and limitation of the approach will be discussed to suggest further improvement.

1. **METHODOLOGY**

In this paper, we will discuss 2 different approaches, pretrained model and self-devised model.

1. **Pretrained models**

Regarding the pretrained model, in this paper, state-of-the-art models with masked word prediction objective is chosen, namely BERT (which is trained with masked word prediction and next sentence prediction objective) and RoBERTa (which is trained with only masked word prediction objective). This is to compare whether removing the next sentence prediction task really results in better performance, as claimed by RoBERTa authors.

* 1. BERT (Bidirectional Encoder Representations from Transformers)

Created and published in 2018 by Jacob Devlin, BERT has risen to be consider the state-of-the-art model for almost every NLP (Natural Language Processing) task. The superiority of BERT comes from not only its architecture, but also from the way input is represented and the objective on which it is trained. Regarding the architecture, BERT consists of multiple Transformer block, which adopt the novel multi-head self-attention, multi-head attention and residual connection (Devlin, et al., 2019). This has allowed BERT to be trained much faster than Recurrent Neural Network and generate much more meaningful embeddings. To create an input to BERT, a sequence is first tokenized using wordpiece (Schuster & Nakajima, 2012), special tokens like ‘CLS’ (indicating the start of the sequence) and ‘SEP’ (indicating the end of a sentence) are also added. After tokenization, tokens are transformed into 3 embeddings: token embeddings, segment embeddings and position embeddings and BERT use the sum of these 3 embeddings as inputs to the model. This has allowed BERT to consider the meaning of the tokens, the position of tokens in relative to each other and the components in sentences. However, the factor that really makes BERT stand out is its modelling objectives: masked word prediction (MWP) and next sentence prediction (NSP). In MWP, 15% of the input words are chosen for masking, and 80% of these are actually masked, 10% of these are replaced with random words and the remaining is kept the same. This is to ensure that the model has enough context to train. Masking is done in a static manner in BERT, since it is generated during data preprocessing step. Whereas NSP enables the model to understand sentence relationship, MWP has allowed the model to fuse the left and right context when generating representation, making BERT better than other models. Finally, regarding the training data, BERT is trained using a total of 3.3 billion word from 16Gb of data (BookCorpus andWikipedia). In this paper, 4 versions of BERT are considered: BERT base (12 Transformer blocks, hidden size of 768 and 12 self-attention heads), BERT large (24 Transformer blocks, hidden size of 1024 and 16 self-attention heads), each with uncased or cased text.

* 1. RoBERTa (Robustly optimized BERT approach)

RoBERTa, proposed by Facebook AI in 2019, resembles most of the features of BERT, except from tokenization step, masking method and model objectives (Liu, et al., 2019). Specifically, in tokenization step, instead of using wordpiece like BERT, RoBERTa uses byte-level Byte-Pair Encoding (BPE) (Sennrich, et al., 2016). Byte-level BPE extends BPE by using bytes instead of characters as the subword units, allowing it to learn a modest size vocabulary that can still encode any input text without introducing any “unknown” tokens (Liu, et al., 2019). Regarding the masking method, instead of static masking, RoBERTa use dynamic masking, which generates a random mask every time a sequence is fed into the model. This has leveraged the need to duplicate the data to avoid using the same mask for each record in every training epoch. RoBERTa also removes NSP and only uses MWM as its modelling objective as the authors has shown that using a different method of sampling training data can help the model perform even better when removing NSP. The data used to pretrain the model is also much bigger than the data used in BERT. RoBERTa uses training data of 160Gb, including BERT training data and additional 144Gb of data (CC-News, OpenWebText and Stories). The model is also trained for longer and in larger batches, which are also believed to contribute to the success of the model over BERT. In this paper, 2 variations of RoBERTa is used, RoBERTa base and RoBERTa large.

* 1. How pretrained model is applied in this paper

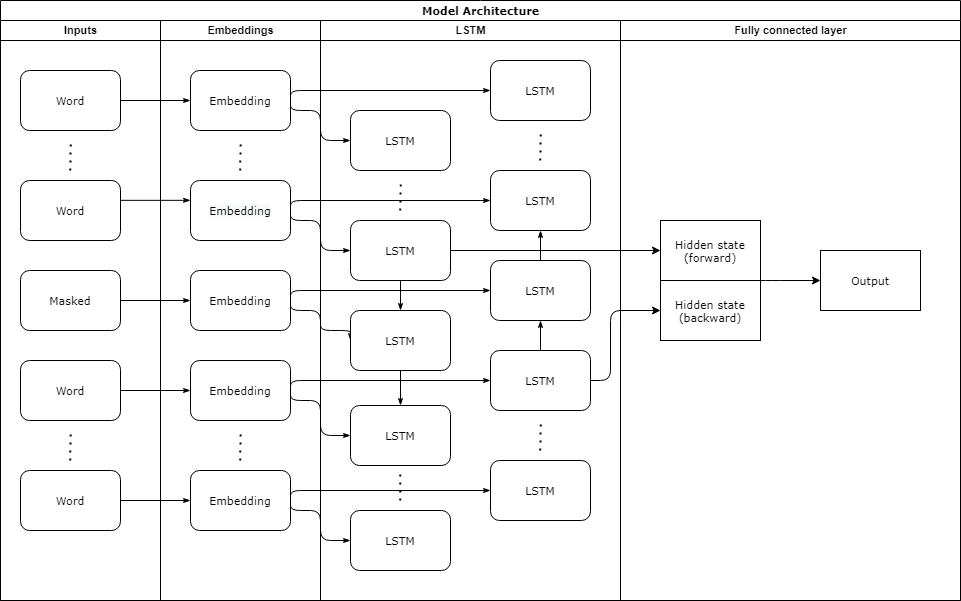
To generate the answer for the multiple-choice question with the pretrained model, the tokenized question is fed into the model to get the probabilities of fitting into the masked position of every word in the vocabulary. The 5 options are also fed into the model’s tokenizer to get their ids. If the option does not exist in the tokenizer vocabulary, it is separated into components and ids for each component are retrieved. For example, ‘instantaneously’ is not in the BERT tokenizer vocabulary, as a result, it is separated into ‘instant’, ‘##aneous’ and ‘##ly’. By the same token, 5 options [‘crying’, ‘instantaneously’, ‘residing’, ‘matched’, ‘walking’] are transformed into 5 lists of ids [[6933], [7107, 17191, 2135], [7154], [10349], [3788]]. Then, in order to choose which option is the most suitable for the masked position, 4 different approaches are considered, as listed in Table 1.

Table 1: Option choosing approaches with Pretrained models

|  |  |
| --- | --- |
| **Option choosing approach** | **Details** |
| Base only | Only consider the first (base) token of every option word and choose the token that has the highest probability of fitting into the place. |
| All with max | Take maximum of the embedding of every token of the option words as the option embedding (not including position embeddings and token type embedddings).  Choose the option which is the most similar to the word that has the highest probability of fitting into the place (in terms of cosine similarity between their embeddings). |
| All with min | Same as above, but take minimum instead of maximum |
| All with mean | Same as above, but take minimum instead of maximum |

1. **Self-devised model**

To tackle this problem by Microsoft, I have also devised my own neural language model. In particular, Bidirectional Long Short-term Memory (BiLSTM), together with a fully connected layer is used. The model architecture can be portrayed as below:



* 1. Model architecture explained
     1. *Inputs representation*

To prepare the inputs to the model, a sequence of text is first preprocessed by performing Named Entity, number and case normalisation. In Named Entity normalisation, all the words that are parts of an entity is replaced with the entity’s category itself. For example, ‘Manchester United’ will be replaced with ‘ORGANISATION’. This helps mitigate the probability of getting out of vocabulary word, and in the same time, boosts the meaningfulness of the input embeddings in this task. These are done in current best performing systems (NLTK and Spacy). The sequence is then tokenized using the same system. Due to the lack of massive training data, fine-tuning the model on top of pretrained word embedding models are preferred. To be more specific, the tokens retrieved are then passed through either Glove (Pennington, et al., 2014), Word2Vec (Mikolov, et al., 2013) or FastText (Bojanowski, et al., 2017) to retrieve its embeddings with the size of 300 (for all three models) or smaller (for Glove – it can be 50, 100 or 200). To determine embeddings of the token, first the token is checked as is, if it exists in the pretrained model’s vocabulary, then embeddings is directly extracted; else the lower-cased version of the token is considered. If it is still missing from the vocabulary, the embedding of the token is initialized as a zero array.

* + 1. *Forward pass*

After extracting input representations, the token’s embeddings are then fed into multi-layer BiLSTM with the goal to extract the representations of the token in the masked position by incorporating both left and right context together. To be more specific, the hidden state of the last layers at surrounding locations of the masked token from respective LSTM directions is concatenated and used as the representation of the masked token. For example, in a sentence ‘My name \_\_\_\_\_ Hieu’, the masked position is 3 (index 2), so we will concatenate the hidden states of the last layers at index 1 from the forward LSTM and at index 3 from the backward LSTM. The output of this stage will be of size (batch size, word embeddings size \* 2).

Then it is activated by a ReLU layer to obtain non-linear property and then passed into a fully connected layer with the number of nodes equals to the word embeddings sizes.

* + 1. *Getting the prediction from the model*

After getting the output of size (batch size, word embeddings size), with the representations of the 5 options extracted through pretrained word embeddings model, cosine similarity between each option and the model output is calculated. The prediction is obtained by choosing the option that has the highest similarity to the model output.

* 1. Training the model
     1. The training data

The training data is prepared using the Holmes data, which includes 523 documents. As questions in the Microsoft sentence completion challenge data mostly consists of a single sentence, the documents in the training set is segmented into multiple sentences using one of the above-mentioned preprocessing systems (NLTK and Spacy). For each sentence, the same preprocessing procedure above is applied, and a random token is masked. Short sentences (with less than tokens) are then filtered out. Next, the original word (before being masked), together with 4 other words randomly sampled from the English vocabulary proposed by NLTK are packed together to create 5 options for the sentence’s masked token.

* + 1. *The model’s training process and objective function*

For each training sample, we define 3 key elements: question (q) and positive (p) & negative (n) answer, in which question is the original sentence that is used to fed into the model, positive answer is the true token in the original sentence that is masked and negative answer is the word that is randomly sampled from the 4 remaining option. Following the same loss proposed in (Tan, et al., 2015), I define the loss function of the model as follow:

where M is constant margin.

The hyperparameters investigated in the training process include:

Table : Hyperparameters investigated

|  |  |  |
| --- | --- | --- |
| **Hyperparameters** | **Group** | **Values** |
| Learning rate | Optimisation | 1e-4 |
| Batch size | Optimisation | 64 |
| Margin | Optimisation | 0.2 |
| Preprocessing system | Architecture | ( NLTK Spacy ) |
| Embedding model – embedding size | Architecture | Word2Vec – 300  FastText\_simple – 300  FastText\_en – 300  Glove\_6B – 50  Glove\_6B – 100  Glove\_6B – 200  Glove\_6B – 300  Glove\_42B – 300  Glove\_840B – 300 |
| LSTM hidden state size | Architecture | ( 64 256 ) |
| Number of LSTM layers | Architecture | ( 1 3 ) |

To exhaustively search an optimal combination of hyperparameters, more than 800 models must be fitted, which is not feasible given strict deadline. As a result, only hyperparameters related to model architecture are examined. Batch size is fixed at 64 because that is the maximum amount of data that can be fitted into the GPU, and some argue that large-batch training can bring about advantages when learning rate is increased appropriately (Liu, et al., 2019). The margin is also fixed at 0.2 as suggested by the paper on QA-LSTM (Tan, et al., 2015).

1. **PERFORMANCE EVALUATION & ERROR ANALYSIS**
2. **Models’ performance**

The performances of the models are evaluated by using accuracy score, which is the ratio of questions with correct answer over total number of questions. Table 3 summarized the models’ performances, together with the hyperparameters/architectures that results in the best accuracy.

Table 3: Summary of Performances

|  |  |  |  |
| --- | --- | --- | --- |
| **Group** | **Model** | **Specifications** | **Accuracy (%)** |
| Baseline | Unigram | None | 27.31 |
| Bigram | Left context only  Probability back-off to unigram | 28.37 |
| Pretrained | BERT | BERT large uncased  Result choosing method: Base only | 78.85 |
| RoBERTa | RoBERTa large  Result choosing method: Base only | 76.73 |
| Self-devised model | BiLSTM | Batch size:64 | Embd model:glove | Embd name:6B | Embd dim:200 | Hidden dim:256 | LSTM layers:1 | Training ratio:1.0 | LR:0.0001 | Margin:0.2 | Preprocessing:nltk | 31.15 |

For reference, unigram and bigram language model’s performance on the test set is also included. As can be seen in the table, BERT performs the best, followed by RoBERTa, whose accuracies are more than 2 times higher than the accuracy in unigram and bigram model. This gap in accuracies can be attributed to the amount of data used for training. Another reason is that BERT and RoBERTa adopt transformers in their architecture, making it much more complex and efficient than simple unigram, bigram model or even LSTM. Specifically, BERT and RoBERTa are considered bidirectional language model (can ‘look both ways’), making the word embeddings generated much more meaningful. Another reason that these pretrained model performs better is the amount of data that they are trained on.

For BERT and RoBERTa, using ‘base only’ approach (explained in section 1.3) to choose the answer results in the highest accuracy. Thus, contextualized embeddings are essential in depicting the correct answers. For BiLSTM, given that the question in the test set is somewhat short, some architecture choices led to worse performance on the test set, namely higher number of LSTM layers, smaller LSTM hidden state size. Moreover, using Word2Vec as the embedding model gives us substantially worse result, in both training and testing phase. For more details, please refer to the tensorboard data attached to the submission. The accuracy for every tested model is pretty good on training and validation set, ranging from 80% to 95% and there was no sign of overfitting detected. However, when applying it to the test data, the accuracy plummeted. This is likely to stem from the way the training, validation and testing set was generated. In particular, whereas in testing set, sentences and options are carefully chosen on the basis of infrequent words, training and validation data is generated using every sentence possible, random masking and random choices from the vocabulary.

1. **Error analysis**

After conducting error analysis, one can realize that BERT performs better than RoBERTa in this particular data even though RoBERTa is trained on more data because BERT (with wordpiece) is better in dividing out-of-vocabulary words into sub-words than RoBERTa (with byte-level BPE). For example, the word ‘quarreled’ is divided into [‘quarrel’, ‘##ed’] with BERT, whereas with Roberta, it is divided into [‘qu’, ‘arre’, ‘led’], which make the base only option choosing approach fails. The problem is exacerbated by the mechanism in which the options in the challenges generated (choose words with low frequencies) (Zweig & Burges, 2011). However, more training data indeed helps RoBERTa outperform BERT in cases where the correct answer is in vocabulary or properly divided.

Another source of errors is from the answer options. Particularly, these are questions that have multiple correct answers. Take the following for example:

* Question: It contained a loaf of bread , a tinned tongue , and two \_\_\_\_\_ of preserved peaches.
* Options:
  + pairs
  + books
  + bottles
  + drops
  + tins (correct answer)

In this example, both *bottles* and *tins* are suitable for the masked position. And while the correct answer is *tins*, the option chosen by the models are *bottles*, making the models fail in this question.

1. **CONCLUSION AND SUGGESTION FOR FUTURE WORK**

With their exceptional performance, BERT and its variants is really the go-to model for mostly every NLP task nowadays, even for a challenging task like Microsoft Sentence Completion Challenge. For these models, I find that adding more data to the training process can benefit down-stream tasks. Choosing the right tokenizer can also lead to a boost in performance. The analysis has shown that wordpiece tokenizer is still probably the best, as it is able to disintegrate word into more meaningful sub-words than other tokenizers.

The analysis does have some limitations that can be overcome to achieve better result. Specifically, considering the first sub-word a proxy for the out of vocabulary word is not always correct. Applying lemmatization or stemming to these words to remove prefixes and suffixes can also be considered. Since the questions in the test data are taken from novels and fine-tuning on data within the same domains is also suitable and can lead to higher accuracy. For BiLSTM, one can improve the performance by replicating the same data preparation procedure as in the original paper (Zweig & Burges, 2011) to close the gap between training/validation and testing performance. Further preprocessing the data with lemmatization or stemming before feeding it into the model is probable too, since the original options and words in the questions are likely to be out of vocabulary word.

**APPENDIX A: Predictions of the models**

Predictions and performance can be found in the file *‘Minh Hieu Tran – predictions.xlsx’*

**APPENDIX B: Detailed performances of the models**

For pretrained models, the result can be found in the file *‘pretrained.ipynb’*

For self-devised models, the result can be found in the folder *‘runs’*. To view this, please use tensorboard

**APPENDIX C: Detailed code**

For pretrained models, the result can be found in the file *‘pretrained.ipynb’*

For self-devised models, the code to prepare training and validation data can be found in the file *‘training\_preparer.py’*. The code for the model can be found in the file *‘model\_fitting.py’*. The hyperparameters search code can be found in the file *‘hyperparam\_search.sh’*

# REFERENCES

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T., 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics,* Volume 5, pp. 135-146.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL-HLT.*

Keskar, N. S. et al., 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. *ArXiv, abs/1609.04836.*

Liu, Y. et al., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692..*

Mikolov, T. et al., 2013. Distributed Representations of Words and Phrases and their Compositionality. *NIPS.*

Pennington, J., Socher, R. & Manning, C. D., 2014. *GloVe: Global Vectors for Word Representation.* Doha, s.n.

Schuster, M. & Nakajima, K., 2012. *Japanese and Korean voice search.* Kyoto, IEEE, pp. 5149-5152.

Sennrich, R., Haddow, B. & Birch, A., 2016. Neural Machine Translation of Rare Words with Subword Units. *ArXiv, abs/1508.07909.*

Tan, M., Xiang, B. & Zhou, B., 2015. LSTM-based Deep Learning Models for non-factoid answer selection. *ArXiv, abs/1511.04108..*

Zweig, G. & Burges, C. J., 2011. *The Microsoft Research Sentence Completion Challenge,* s.l.: Microsoft Research Technical Report MSR-TR-2011-129.