Microsoft Research Sentence Completion Challenge

Advanced Natural Language Processing (968G5):
Assessed coursework 1

1 Introduction

This coursework is based on assessing and evaluating the different models on Microsoft Research Sentence Completion Challenge (MRSCC) [1]. This report has used the data set available from the lab. The moot point of this report is to provide and discuss the different ways a model can perform well and may differ depending upon some circumstances.

1.1 Microsoft Research Sentence Completion Challenge

This challenge came out from the observation of Microsoft's people after seeing that there is a dataset available to practice the natural language semantic relationship using different types of modelling [1]. Being a curious mind, the people from Microsoft thought to discuss further and find a way to solve and estimate the semantic relationship between the text. This challenge has more than thousands of sentences. Moreover, they have been taken from the famous Sherlock Holmes novels. In a total of 1040 sentences, a word is missing in each of the sentences to make out the meaning of the sentence also there were five choices for them to fill it up in these blank spaces. A state of art method of n-gram model has been used to find the most probable word for these sentences [1]. But there was a risk of selecting the right word among them, neglecting the incorrect one instead. Initially, the n-gram was used to train the 540 texts from the Gutenberg corpus [1]. The word choosing process stuck to different patterns and parameters also, these all words have almost equal probability with the selected word. Initially, humans have started to pick the wrong choice out of these five given words, without making a clear assumption of picking up the correct choice for the sentence.

1.2 Motivation

The motivation of this challenge is always designing a natural language model which can predict the best word out of the given ones. The Data set has not been a well-structured. It has a lot of loopholes, which become a stumbling block on the way of the model to train and test the sentence. We can change the parameter while performing a model to the sentence completion task.

2 Background

Here there is a discussion about the background information to help in more understanding about the different models that were used in the report. A bit brief about the unigram, bigram, trigram and RoBERT-base models are discussed.

2.1 N-gram

The n-gram model defines a rigid approach to building a language model by directly finding the probability of each word and sequences to find the predictions [2]. Also, when we try to get a probability of missing a word in a given sentence an n-gram model will calculate the more likely sense word to happen based upon the number of chances of this occurring in a training data set. The most common n-gram model is the unigram model. This has a condition like the most likely word can be chosen by just picking the most likely word to happen in the data set.

$$P(w_j) = \frac{w_j}{\sum w_N}$$

In other meaning, the probability of a word happen is equal to the frequency of the text in the data set (w_j) divided by the sum of the all-frequent words in data set (w_N). This is the one of the simplest methods for getting the n-gram model probability. It is hypothetically believed that the probability of a favourable word is not dependent upon the other words examined in a sentence. Whereas, bigram includes the previous work to the sequence for finding the probability of the current word.

$$P\left(\frac{w_j}{w_{j-1}}\right) = \frac{(\frac{w_j}{w_{j-1}})}{\sum(\frac{w_N}{w_{j-1}})}$$

In n-gram modelling, we can consider n number of words. Hence it is defined as an n-gram model. In order to apply these four types of n-gram model over Microsoft data the probability of the candidate words in the sentence was measured to the training data from the novels.

2.2 BERT

It is the bidirectional encoder representations from transformers that has affected the performance on a lot of different natural language tasks [3]. This model is mostly related to transformer encoders [4]. These types of encoders are consisting of two types, multi and feed forward neural networks. The BERT base model is a group of 12 transformer encoders [3]. These give a way to transform input is a conclusive way to learn relationships between the encoders.

2.3 RoBERTA

RoBERTA is a robustly optimised bidirectional encoder representations from transformers approach are a version of BERT [5]. It happens that BERT is not trained for a longer time so that RoBERTA is introduced by revising the BERT on larger batches and more text. They only depend upon masked language modelling tasks. The RoBERTA models provide a robust masking where a lot of different training data have different masks in every training epoch. Their architecture modelling is quite similar, still having a lot of differences in features. RoBERTA is better for inner representation of language as compared to BERT. RoBERTA has outperformed the BERT in most of the challenging tasks in language processing.

2.4 RoBERTA-BASE

This is a pre-trained model on natural language using masked language model. It is case sensitive model. It works mostly on large corpus in a self-supervised way. It is a bit different from the other MLM model, like data used for pretrained and number of trainings cross the data [6].

3 Method

There were six types of distinct methods that were applied to the historical results [1]. The best method was to pick up the human. It has been considered that the human method has outscored the other five one. After the human one, it was the latent semantic method finding the similarity between the vector type of different text opposite to the different word. This LSA method has given the forty nine percent accuracy of answering the correct questions. Which was the highest among the computational methods. The rest four methods were based on n-gram modelling. All the n-gram models have accuracy in between 28% to 40% [1]. This report has used two different methods to examine the

investigation of the report. First one is the n-gram method.

Model	Accuracy (%)
Bigram	18
Trigram	19
4-gram	45
Unigram	20
RoBert-base	80

Table.1 Results of each model for a given data set.

Where bigram, trigram and 4-gram were used for n-gram modelling. The second method was RoBERT-Base masked language model. Both n-gram and MLM have shown their accuracy on the different set of test sentences. The second method is quite popular, the hugging face model. It should be considered that the RoBert-base method is of the BERT model which is pre-trained on tasks like masked language. These pre-trained models are accessed through hugging face libraries.

We have used simple data pre-processing using the library to take out the headers of questions and answers from the Gutenberg. The RoBERT model was implemented with the use of Pytorch library [7]. Also, the hugging face library was used on google colab using the GPU [8].

3.1 RoBERT – BASE MODEL

The model has been used with a hugging face library. Which helps in transforming the training-based model [9]. Also, I have used the data set application for this task. This model comes with a pre-trained on very large text corpus. Meanwhile, it is suggested that this model has better performance with downstream task. This means, it is retrained with a masked based language task with a training text of the late nineteenth century. Then we trained this model; we check its performance with Microsoft data in the retraining phase. By the analysis of the report, it was considered that this model has outshined the other model (n-gram) with its performance in this task.

The task started with pre-processing task in which the training text was tokenized using the hugging face tokenizer. As soon as we converted them into the token ID, they were further divided into a series of tokens. We also use the loss function to find the maximum changes in between the true and predicted values. In general, there were two plans used for inference time. Initially, it has predicted a mask text token in position of missing word in each of the given sentences. This is mostly used for getting the probability for each candidate token. The highest one has chosen according to the prediction strategy. The other strategy in this model was to take care of this task in different way: first it gives a sentence to each word and the sentences are fragmented into distinct token Id. One single word can be divided among different tokens (like ‘experimental’ is divided among into ‘exp’, ‘periment’, ‘ental’, ‘mental’). Henceforth, the divided tokenized texts are then masked with each of the given token ID. Like, if the word is of 4 tokens in total length, then the candidate sentence has 4 different mask tokens in their respective places. Each token was predicted and final prediction were taken by taking their mean. This task in a bit cumbersome but it has better result than the prior one.

The RoBERT base model is extensively large and therefore it is quite expensive. It was observed that it almost took four to five hours to run the entire training data using the GPU.

The hyper parameter which was used for this model has helped the accuracy of the model. Mostly, Google Colab has been used for this modelling and batch size and sequence length was set to a particular point before GPU running time out. A few numbers of important hyper-parameters were set to inquire the difference in the training trial. The model was used with

training data set, holdout accuracy, batch size, learning rate and number of epochs.

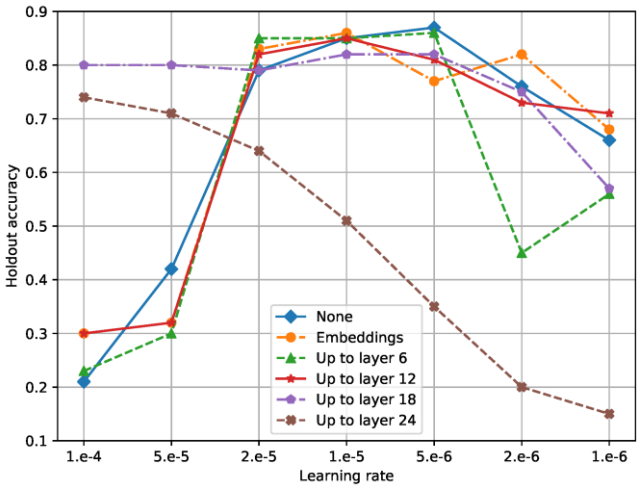

Fig.1. Holdout accuracy with learning rate for BERT model.

3.2 N-Gram Model

The second method was n-gram method, where we investigated the model accuracy with bigram, trigram and 4gram.

The use of the n-gram model for sentence completion is quite significant. This model creates an n-gram system. This system works out for each of the text-words in the whole corpus and builds a word of the dictionary to also create the probability for the upcoming word. A unigram system works differently, it just finds the probability of a word occurring in the corpus and divided by the sum of all the words present in that corpus. bigram searches for the one word to the left side so it creates a dictionary based on this way. In the example “the man sat on the sofa”. A bigram will check the word ‘the’ and then look for the word ‘man’ and ‘sofa’ in the different position. The dictionary would build up probability for the word ‘the’ with respect to these two words. As soon as we get more data, the probability keeps on changing for the n-gram.

PERPLEXITY

It is the one of the parameters which help finding the best model for n-gram modelling. It is the multiplicative inverse of the probability given to the test set by the n-gram model. The perplexity parameter was found for the training data set. The perplexity was examined for bigram, trigram and 4gram. It was found that as we increase the data set the perplexity keeps decreasing. So, the less perplexity the more probability for that dataset.

N-Gram	Perplexity		
	8	20	40
Unigram	107.0178	116.2954	221.4313
Bigram	51.6554	56.5050	74.3852
Trigram	37.5881	25.3478	17.0038
4-gram	12.3239	9.2456	6.4820

Fig.2. Perplexity of different n-gram models.

We can observe from the fig.2 as soon as we take the twenty percent of training set the perplexity changes. The better perplexity is found with 40% of the training data set. The trend gets better as the n-gram size increases.

Fig.3. seems the possibility of known parameters with the test set perplexity. There are few fixed vocabularies which are unchanged from training and testing data. Fig. 3 depicts that perplexity is decreasing as soon known values increase.

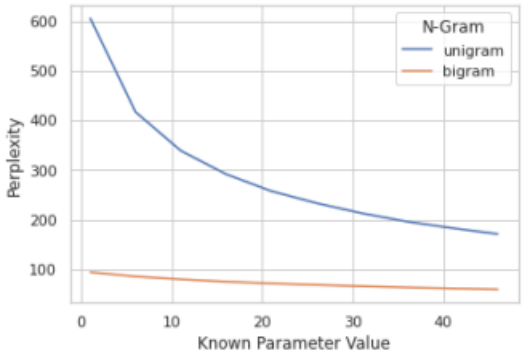

Fig.3. Perplexity with known parameter value

ANALYSIS

As we build the bigram model in the lab, in this way we build the trigram and 4-gram model. In trigram, we build a dictionary which gives us one more dictionary to find the second candidate word. This dictionary backs this another candidate word by getting the probability. Some questions in the data set have very straight forward answers but while applying the model to them the accuracy was low with smaller n-gram.

The quadrigram was constructed upon the trigram by just adding another layer on the trigram. Which permits the third word to be added to the dictionary. This has helped further narrowing down the possibility to the chosen words, which definitely increases the accuracy of the system. There is a chance that few of the grouping of phrases and target words will not seem in the more n-gram model. Which will in return decline the accuracy of the model.

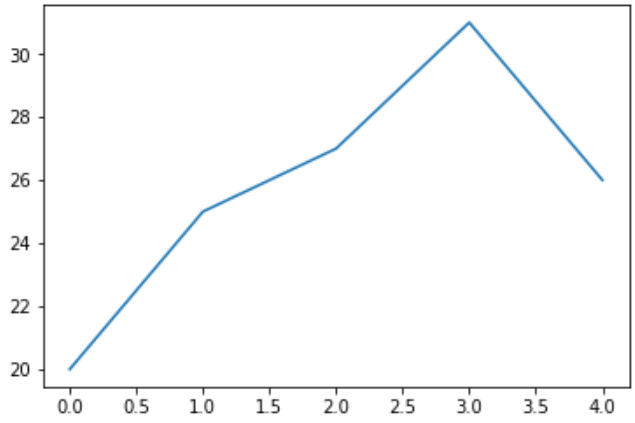

Fig.4 N-gram model accuracy

In fig.4 we can see the accuracy of the n-gram model increases from the n=0 for random value to n=1 for unigram. The changes appear to a certain point. The accuracy of this model gets its maximum point at 32% for trigram. For bigram the highest accuracy gets around 28%. The accuracy after trigram gets declined. Hence the n-gram model is not perfect for this data set. It eventually declined with adding more combinations.

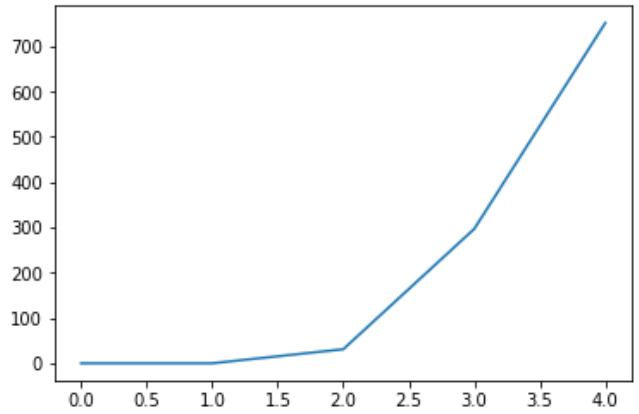

Fig.5 Number of error in n-gram system

The fig5 shows that there are a lot of problems and challenges in sentence completion task that the n-gram system was not able to get the optimal solution. Most probably its of the context word and the target word were not present in the memory. It can be observed that bigram even starts getting failed at mixing the words. This is most likely because the sentences contain a lot of different type of phrase context. These phrase contexts may be proper noun which can appear in the training data.

Even quadrigram was having more than 750 errors. To test on these problems, it was witnessed that 4gram was able to get 45% accuracy when it was able to find the solution. It was also checked that bigram was able to find solution in a tricky phrase, whereas the quadrigram which has a bigger context window was less accurate to find solution with very less-common phrases and their combination.

This type of error can be taken away with stacking the n-gram models altogether. The accuracy might be lower for lower n-gram models but unfound the solution was also lower for them.

4 Evaluation

The Microsoft research sentence completion challenge is an equally concentrated multi-level class challenge. It means, a model which abruptly

can take answers can get almost 20% accuracy on a given task.

In this report, we applied the two important natural language models. It was found that the n-gram model has performed very poorly on this task of Microsoft sentence completion challenge. Whereas, the RoBERT model has the highest accuracy with 80%. The n-gram was not that accurate due to its selection of candidate words for different sentences that were taken via the application of the naïve n-gram model. Improvement is required for this method to test its hyperparameters and data pre-processing algorithms. The simple n-gram model which had not much pre-processing used has scored better with a pre-processing model. N-gram systems are usually sensitive to out of vocabulary words and its major success was how the not known words were taken care of through the known metric. We have already seen that increase in size of words and n-gram leads to less accuracy as well as perplexity.

Also, the transformer base BERT model has better performance on language modelling. Its ability to perform well for this type of challenge is due to its ability of variable series width, better contextual learning and large pretrained data was the cause of its successful on this task [5]. They are pretrained with various large corpus even they performed very well on novel-based task. The hyperparameter like training time and batch size provided the better result for this model.

5 Further Work

This section talks about the major development to be made on the applied models and ideas to further work in this field. It has discussed both the model in length and breadth, both their scores are acceptable. Creating a perfect balance between the size of the training set and the coping with the unknown can enhance the probability and the accuracy. To check the statistical application, a one tail hypothesis can be run on the result of the ensemble approach to find the unlikelihood that its accuracy was due to random frequency. Further application of the smoothing method can be utilized on the n-gram system to enhance the accuracy. BERT model with increasing inference time and optimisation can take the accuracy to a higher point [10]. Also, a lot of research to be further done to work properly on the BERT model (base and distilled).

6 Conclusion

The Microsoft research data has given a cumbersome work for the natural language scientist. Both methods have played pivotal roles in studying and assisting the given challenge. Models have predicted an accurately most probable word for each of the given sentences. Candidate text was chosen by the BERT model via a probabilistic model, which can work outside a generative probabilistic model. Trained model has outshined the other model by using a model mechanism which has the ability to operate covariant sequences.

References

[1] G. Zweig and C. J. C. Burges, “The microsoft research sentence completion challenge,” 12 2011. [Online]. Available: http://research.microsoft.com/scc/

[2] J. Weeds, “Anle university of sussex week 2 slides,” UoS Lectures, pp. 11–12, 2021.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, vol. 1, pp. 4171–4186, 10 2018. [Online]. Available: http://arxiv.org/abs/1810.04805.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Lukasz Kaiser, and I. Polosukhin, “Attention is all you need,” vol. 2017- December. Neural information processing systems foundation, 6

2017, pp. 5999–6009. [Online]. Available: https://arxiv.org/abs/1706.03762v5

[5] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," arXiv, 7 2019. [Online]. Available: http://arxiv.org/abs/1907.11692.

[6] J. Alammar, "The illustrated transformer – jay alammar – visualizing machine learning one concept at a time." 6 2018. [Online]. Available: http: //jalammar.github.io/illustrated-transformer/.

[7] Pytorch, "Word embeddings: Encoding lexical semantics — pytorch tutorials 1.8.1+cu102 documentation," 2021. [Online]. Available: https://pytorch.org/tutorials/beginner/nlp/ word embeddings tutorial.html

[8] Google-colab, "Google colab - cloud computing." [Online]. Available: https://colab.research.google. Com.

[9] Hugging-Face, "Hugging face library." [Online]. Available: https://github.com/huggingface

[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," arXiv, 10 2019. [Online]. Available: http://arxiv.org/abs/1910. 01108