Microsoft Sentence Completion Challenge

1. Introduction

Text prediction algorithms for automatic or semi-automatic sentence completion have been vastly discussed in the Natural Language Processing domain. (Spiccia, Augello, Pilato, & Vassallo, 2015)These algorithms are widely used to enhance communication speed, reduce the time taken to compose the text, and play a pivotal role in developing cognitive abilities to intercept and examine written texts. This assignment focuses on building and evaluating models for Microsoft Research Sentence Completion Challenge and carries out a comparison of the effects of these models on the performance. The task of sentence completion, which aims to infer missing text in a given sentence was carried out to access the reading comprehension ability of machines using different algorithms. (Hassan, Sanchez, & Domingo-Ferrer, 2020)

1.1. Sentence Completion Challenge

Sentence completion challenge is a type of cloze-style task whose goal is to choose a correct word from a given choice of options to fill in the blank in the question sentence. This class of questions is useful to understand abilities like linguistic proficiency, common knowledge, and logical reasoning at different levels. The problem statement is a part of the Microsoft Research Sentence Completion Challenge where the authors observed that there are few public datasets available for semantic modelling which can be used to evaluate the performance of their systems in the task of modelling semantic meaning in the texts.

The Sentence completion challenge data is constructed from Project Gutenberg data. 1040 seed sentences were selected from five of the Sir Author Conan Doyle's Sherlock Holmes novels. In each sentence, a low-frequency focus word was selected and thirty alternative words were generated using the maximum entropy n-gram model. The n-gram model was trained on 540 texts from the Project Gutenberg collection where most of the selected texts were from 19th-century novels. Out of these thirty words, four best imposter words were selected. All the imposter words had similar co-occurrence statistics with the original words. Each of the four imposter words was hand-picked from a large set by human
judges whose role was to pick sentences where the generated words fit best, without making the original correct answer less clear. (Zweig & Burges, 2011)

1.2. Motivation

In the assignment, two approaches – n-gram modelling and BERT for evaluating the models on the test sentences were used. The n-gram model is extensively used in text mining and Natural Language Processing. The other model used is Bidirectional Encoder Representation from Transformers (BERT) which has caused a stir in the Natural Language Processing community by presenting state-of-the-art results in a wide variety of NLP tasks like question answering, natural language inference, etc. BERT's key technical innovation is applying the bidirectional training of Transformer to language modelling. The motivation behind applying these models to the sentence completion challenge data was to evaluate the performance of the n-gram model and BERT and explore the effects of applying different features like smoothing and hyperparameter tuning in the development process and gain insights into how the models process natural language.

2. Methods

The MSR SCC paper has details about the analysis of 6 different approaches. The highest scoring was the human baseline. Latent Semantic Analysis (LSA) using similarity calculations between the cosines of the angles between vector forms of different words and the candidate vector gave the accuracy of 49%. The other approaches were variations of n-gram language modelling which gave the accuracy in the range were variations of n-gram language modelling which gave the accuracy in the range 31% to 39%.

Method	Accuracy
Human	91%
Generating model	31%
Smoothed 3 gram	36%

Smoothed 4 gram	39%
Simple 4 gram	34%
Latent Semantic Analysis	49%

Figure 1: Accuracy obtained

The results from these models indicated that the "Holmes" sentence completion set is indeed a challenging problem with a level of difficulty comparable to SAT questions. Simple models such as n-gram do not give the best results.

In this investigation, two different approaches are used to analyze the problem. The first approach is using n-gram modelling. The n-gram modelling was tried with simple unigram, bigram, and trigram models as well as with smoothed unigram, bigram, and trigram models. The smoothed n-gram model shows some progression in the accuracy than the simple n-gram model after adjusting the hyperparameters. The second model used is BERT which is considered to give state-of-the-art results. After applying the BERT model, the accuracy showed a positive increment as the model could move across the local information constraint imposed on n-grams and provide global semantic coherence.

2.1. N-gram Language Model

Given a sequence of N-1 words, an N-gram model predicts the most probable word that might follow this sequence. The N-gram model is built by counting how often the word sequence occurs in corpus text and then calculating the conditional probabilities. The model uses the chain rule of probabilities, maximum likelihood estimation, and Markov assumption to generate the next likely word. A model that relies on how often the word occurs without looking at the previous words is called unigram. If a model only considers the previous word to predict the current word, it is called bigram. If the model considers two previous words to predict the current word then it is a trigram model. The N-gram language model assumes that the next word depends only on the previous N-1 words and the probabilities can be estimated by normalizing the N-gram counts. N-gram modelling has a wide range of applications from automatic spelling checks and semantic analysis to speech recognition. As the model has shown successful results in various fields, N-gram modelling was selected for the sentence completion challenge task.

2.2. BERT

Bert makes use of Transformers, an attention mechanism that learns contextual relations between words or sub-words in a text. Transformers include two separate mechanisms- an encoder that takes input text and a decoder that gives the prediction for the task. Since BERT is used to generate a language model only an encoder is used. As opposed to the directional model, which read the text sequentially from left to right or right to left, BERT reads the entire sequence at once. This allows BERT to learn the context of the word based on all of its surroundings. When training language models a big challenge is to define a prediction goal. Many models predict the next word in a sequential pattern, which limits the context learning. To overcome this limitation, BERT uses Masked LM. As BERT is bidirectional and can improve the accuracy and give state-of-the-art results, it is selected for the sentence completion challenge task.

3. Experimental Results

3.1 N-gram Language Model

In this model the questions were first tokenized to break the raw text into words and these tokens will be further used to calculate the probabilities of the words. __START and __END was added to each sentence in the dataset with the tokenization step. The N-gram model was trained with an increasing number of text files starting with 10 files in the first iteration, and 20 in the next iteration and so on. For calculating the probability of a word in the given sentence, the model gives the token which has the maximum probability among the tokens in that context. The formula for calculating the probabilities is given below. Here, cnt is the count of word sequences.

$$P(w_i|w_{i-1},\dots,w_{(i-n)+1}) = \frac{cnt(w_{(i-n)+1},..,w_{i-1},w_i)}{cnt(w_{(i-n)+1},..,w_{i-1})}$$

Handling Out of vocabulary words

Out of vocabulary words are the words that are not in the training set but appear in the testing data. When such words occur in the model, the model assigns a probability of zero to that word, as a result of which the likelihood becomes zero. To avoid this issue, the N-gram model handles the OOV words by passing a "known" parameter during the initialization. When the model is initialized, it checks for the words whose occurrence is less than the value of known and replaces it with the "UNK" token. For experimenting, the known parameter was iterated with values of known = 2, 3, and 4. The language model was first trained on 10 files with unknown =2. "UNK" words are the maximum occurring words among the tokens. As the vocabulary size of the files was less, the model did not perform that well with 10 files. As the number of files increased, the performance of the model increased as well.

The probability of a "UNK" word in the unigram model comes out to be 0.01031. The total number of times a "UNK" word has occurred in bigram is 2404 and in trigram is 10521. This will impact the sentence completion because even if the word occurred less time, it might have a probability greater than some other words in the token or UNK.

Calculating Perplexity

Perplexity is a measurement that decides how well the model predicts the sample data. Perplexity calculations were done for all the training files in unigram, bigram, and trigram. As the number of files increased, the perplexity decreased. This suggests that with an increasing number of files the vocabulary size also increases and the probability of the words can be calculated efficiently. But as the vocabulary sizes are different comparing perplexity with different file sizes is not a feasible idea. The perplexity of the N-gram model also showed improvement with higher-order N-grams like bigram and trigram as compared to the unigram model. The texts generated with bigram and trigram showed coherence.

Context

For experimenting with the context, the model was validated by using the left context. This did not change the accuracy obtained for any of the models. When the model was validated with the right context, the accuracy decreased by 2%. After investigating the tokens in the right context, it looks like the tokens in the right context carry less significance in the context as compared to the tokens in the left context.

Evaluation

The accuracy of the obtained by random choice for the sentence completion task by the bigram model is around 19% and the trigram model is around 20%.

Hyperparameter tuning

Kneser-Ney is widely used in speech and language processing and has shown successful performance in the natural language processing domain. Therefore, to tune the model, here Kneser-Ney smoothing is used. Kneser-Ney is evolved from absolute discounting interpolation which makes use of lower-order language models as well as higher-order language models, recollecting some probability mass from higher orders to lower orders.

Kneser-Ney deals with OVV words, by moving some probabilities towards unknown N-grams. It depends upon the idea of continuation probability associated with each unigram.

$$P_{Continuation}(wi) \propto |\{w_{i-1} : c(w_{i-1}, w_i) > 0\}|$$

Therefore, Kneser-Ney can be calculated by the below formula. Here, λ is a normalizing constant.

After applying the Kneser-Ney smoothing to the N-gram model, the accuracy of the model increased by around 5%. The accuracy was also checked by taking OVV thresholds words and the number of files. The table below

describes the accuracy of the Unigram, Bigram, and trigram model after applying Kneser Ney Smoothing. As it can be seen from the table in the unigram model, the accuracy increased by around 2%. With the number of files increasing, and the increase in the OVV threshold there is a slight increase in the accuracy of the Unigram model. The Bigram model shows an increase in accuracy by around 4% to 6%. In the bigram model, an increase in the OVV threshold and the number of files increases the accuracy slightly. With 10 files and an OOV threshold of 2, the accuracy is increased by 4%, and with 100 files and an OOV threshold of 4, an increase of 6% is observed. The Trigram model also shows an increase of 4% to 6% in the accuracy. With 10 files the accuracy for trigram model is 24% and with 100 files the accuracy is around 26%.

	number of files	OOV threshold	unigram accuracy	bigram accuracy	trigram accuracy
0	10	2	0.185577	0.238462	0.235577
1	10	3	0.203846	0.237500	0.246154
2	10	4	0.187500	0.239423	0.240385
3	55	2	0.194231	0.231731	0.233654
4	55	3	0.205769	0.231731	0.232692
5	55	4	0.193269	0.231731	0.232692
6	100	2	0.189423	0.253846	0.253846
7	100	3	0.167308	0.253846	0.253846
8	100	4	0.172115	0.253846	0.253846

Table 2: Accuracy of the N-gram Models

3.2. Bidirectional Encoder Representations from Transformers (BERT)

The BERT has two models, $BERT_{BASE}$ and $BERT_{Large}$. I this assignment, $BERT_{BASE}$ model is used. This model has 12 encoders with 12 bidirectional self-attention-heads and is pretrained from unlabelled data extracted from the Book Corpus with 800M words and English Wikipedia with 2500 M words. In the first step, the texts are lowercased and tokenized using WordPiece and a vocabulary of size 30,000. During the tokenization, BERT uses special tokens [CLS] and [SEP] to understand the input properly. (Kumar, n.d.)The [SEP] token is inserted at the end of each input. This token helps the model to understand the end of each input and the start of new input. [CLS] is a classification token and the last hidden layer of BERT which is used for the classification of the tasks. From the tokens three tensors are generated with token_ids, token_id_types, and attention mask, and these token_ids are converted into labels for training and optimization. The tokens are then masked in input_ids. After masking, the similarity is calculated using cosine similarity. BERT uses feed-forward network, so the backpropagation criteria are removed. The questions and answers are given as input to the encoder. The answers generated are compared with gold standard, as BERT gives state-of-the-art results. The BERT model gives accuracy in the range of 52%-60%. This accuracy is much higher as compared to the N-gram model

Hyperparameter tuning

The hyperparameters that we choose, have significant impact on the performance of the model. BERT is a pre-trained model where the parameters are tuned already. In this assignment, $BERT_{BASE}$ model is used. The difference between $BERT_{BASE}$ and $BERT_{Large}$. Is on the number of encoder layers. $BERT_{BASE}$ uses 12 encoder layes, while $BERT_{Large}$ has 24 encoders. As the number of layers in BERT increase, the parameters increase as well. Fine tuning the BERT model with $BERT_{Large}$ will require a lot of hardware specifications and memory. Therefore, the finetuning of $BERT_{BASE}$model was not performed practically but it has been explored empirically for this assignment.

There are other ways in which the BERT model can be fine-tuned. The first method is to set a Baseline with Grid Search over a set of predefined hyperparameters. The search space recommend by BERT Authors can be used. For this assignment per_gpu_batch_size, learning_rate and num_epochs were used. We can then run the model for a total of 18 trials or full training runs for each combination of hyperparameters to increase the accuracy. The Grid Search method increases accuracy by 5% (Kamsetty, 2020)

The second method is to improve Grid Search with Bayesian optimization. In this approach a Gaussian Process model is fit which predicts the performance of the parameters (loss) and informs future hyperparameters. For this method weight decay and warmup_steps are also added in the grid Search with

per_gpu_batch_size, learning_rate and num_epochs. For the sentence completion challenge a total of 60 trails can be run. This method increases the accuracy by 6%.

4. Analysis

After tuning the hyperparameters, the N-gram model shows improvement. For BERT the hyperparameters were not tuned experimentally but analyzed empirically due to hardware and memory restrictions. It can be suggested that the BERT model will also show a 5%-6% of improvement in the accuracy after tuning the hyperparameter. OOV words also impacted the accuracy of the N-gram model. It was observed that by setting known words to 2, 3 and 4 the accuracy improved in unigram model. In bigram model the accuracy increased by setting known to 2 and 3, but when known was set to 4, the accuracy remained constant. This might happen due to overfitting of the model as the model skipped learning by storing more mass in the unknown token. In N-gram model with increasing size of documents and OOV words, the perplexity decreased. BERT used $BERT_{BASE}$. The accuracy of BERT can also be increased by using $BERT_{Large}$ and making the changes in hyperparameters as discussed above. The table below shows the accuracy of both the models for comparison.

No	Model Name	Accuracy
1	Unigram	0.2390
2	Bigram	0.2538
3	Trigram	0.2983
4	BERT	0.55

Table 2: Accuracy of models with smoothing

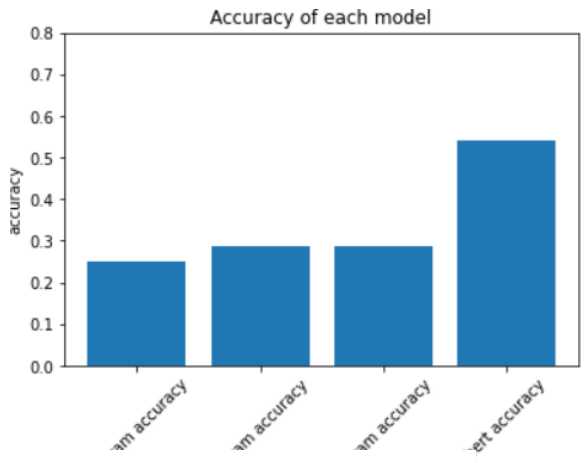

Figure 1: Bar plot of accuracy

From the above graph and table, it is clear that BERT gives the best accuracy between 52%-60% for sentence completion challenge. The accuracy of the trigram model is better than bigram and unigram. Trigram gives accuracy around 24%-30%.

The goal of any language model is to convey information. To measure the average amount of information conveyed in a message, entropy metric is used. It measures how much information is produced on average for each letter of a text in the language. As per the results obtained from calculating entropy, it is observed that Bigram has the highest entropy. The lowest entropy value is in trigram which suggests that trigram can generate meaningful information from the message than unigram and bigram.

No	Model Name	Entropy
1	Unigram	0.750198
2	Bigram	1.495098
3	Trigram	0.135229

Table 3: Entropy of models

The N-gram language model was also evaluated for perplexity. Perplexity is how well a probability distribution predicts a sample. For a model to perform well, its perplexity should be less. The table below describes the perplexity of N-gram model and it can be seen that trigram has the lowest perplexity and hence performs well than unigram and bigram.

No	Model Name	Perplexity
1	unigram	2.1174
2	bigram	4.4597
3	trigram	1.4479

Table 4: Perplexity of models

Conclusion

In this assignment 2 language models were implemented on sentence completion challenge to evaluate the performance and how the models process natural language. From the results obtained from the

models, it can be concluded that pre-trained models with transformers perform well. The N-gram models performs well than by tuning the hyperparameters than by the random choices. The language models are evaluated with metrics such as perplexity, entropy and tuning the hyperparameters. The N-gram model performs well after tuning the hyperparameters. It was observed that adding and tuning the hyperparameters can improve the performance and accuracy of the models.

Future Work

Human level understanding of this task has the accuracy of 91%. To improve the accuracy of models' further work should be with the help of hyperparameters. To improve the performance of the N-gram model, using neural network models like Recurrent Neural Network (RNN) or Long-Term Short Memory (LSTM) can improve the accuracy to a great extent. The BERT model can be improved by tuning the hyperparameters and using the $BERT_{Large}$ model. These few improvements can achieve the desired performance.

References

Hassan, F., Sanchez, D., & Domingo-Ferrer, J. (2020). Utility-Preserving Privacy Protection of Textual Documents via Word Embeddings. *IEEE Xplore*.

Kamsetty, A. (2020). *Hyperparameter Optimization for 🤗Transformers: A guide*. Retrieved from Medium: https://medium.com/distributed-computing-with-ray/hyperparameter-optimization-for-transformers-a-guide-c4e32c6c989b

Kumar, R. (n.d.). *All you need to know about BERT*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/05/all-you-need-to-know-about-bert/

Spiccia, C., Augello, A., Pilato, G., & Vassallo, G. (2015). A word prediction methodology for automatic sentence completion. *IEEE Explore*.

Zweig, G., & Burges, C. J. (2011). The Microsoft Research Sentence Completion Challenge. *Microsoft Research*.