Analysis of N-grams and BERTs for Sentence Completion

By: Joakim M. Torsvik

Date: 19th of April 2022

1.0 Abstract

Sentence completion tools have evolved to become very helpful in writing texts. Smart phones can autocorrect misspelled words and suggest what the next word based on the previous text, e-mail programs give suggestions on how to complete the sentences and grammar programs suggests how to improve one's writing. This paper will analyse how well n-grams and the Deep Learning language model, BERT, can predict missing words from the Microsoft Sentence Completion Challenge [MRSCC] by using training data from Sherlock Holmes novels in the Project Gutenberg corpus.

2.0 Introduction

In 2011 Microsoft introduced the Microsoft Sentence Completion Challenge, which consists of 1,040 sentences with a key word removed from each sentence. The challenge is to create a language model that is capable of predicting which word that is removed, with five possible alternatives given to the challenger where one is correct, and the rest are wrong. Some of the wrong alternatives could easily fit in the sentences and some are obvious wrong answers. The data is gathered from Project Gutenberg's Sherlock Holmes novels, written by Sir Arthur Conan Doyle in the late 19th century to the mid-20s in the 20th century. In order to get as accurate as result as possible, the language models are trained on the very same Gutenberg corpus that the sentences are taken from, however as the baseline model will show, predicting a correct result is still quite challenging.

This paper will look at two model to try to predict the correct result, one being a baseline n-gram model consisting of unigram, bigram and a trigram. The other language model to test is the ground-breaking BERT-model made by Google and an "upgraded" version of BERT made by Facebook, called RoBERTa.

3.0 Methodologies

This paper will cover a baseline n-gram model and pre-trained BERT language model as well as the upgraded RoBERTa-model. This section will cover the theoretical backgrounds of these models.

3.1 N-Gram

N-grams are contiguous sequences of n-length word sequences which captures the most words that are being used together most frequently. Most commonly, n-grams are used as language models for predicting the next word in a sentence by calculating the probability that the previous word(s) are commonly used together with the target word. E.g.: When training a bigram-model on the famous quote in Shakespeare's Hamlet, *"To be, or not to be"*, the model would return an predicted output *"be"* whenever the previous word is *"to"*.

Unigram: $P(w_1) = \prod_{n=1}^{N} P(w_n)$

Bigram: $P(w_1) = \prod_{n=1}^{N} P(w_n|w_{n-1})$

Trigram: $P(w_1) = \prod_{n=1}^{N} P(w_n|w_{n-2},\ w_{n-1})$

In this assignment, the unigram, bigram and trigram have been tested to predict the missing words from the MRSCC.

In order to capture words that doesn't appear in the training data (Out-of-Vocabulary [OOV]), a threshold was set to remove all words that appeared less times than the threshold. The frequency of these words was then stored in a phantom keyword which represented unknown words from the trained language model. Kneser-Ney method for smoothing was also used on the frequency counts of all words in the vocabulary. The Kneser-Ney [KN] smoothing technique evolved from Absolute Discounting to smooth the frequency distributions of their respective n-grams. The Absolute Discount

smoothing technique essentially uses a discount value (λ) which subtracts a small part of the frequency of each word above 0. KN smoothing is an extension of Absolute Discounting where the lower-order distribution that one combines with a higher-order distribution is built in a novel manner. In other smoothing algorithms, the lower-order distribution is generally taken to be a smoothed version of the lower-order maximum likelihood distribution. However, a lower-order distribution is a significant factor in the combined model only when few or no counts are present in the higher-order distribution. Consequently, they should be optimized to perform well in these situations (Stanley F. Chen, 1999). A very famous example is "*The San Fransisco problem*". If the city of San Fransisco is often referred to in the training data, then the unigram probability of "*Fransisco*" will just as high, or higher, than the bigram probability of "*San Fransisco*" and the absolute discounting method will assign "*Fransisco*" with a high smoothed frequency. However, this word shouldn't have such a high stand-alone frequency because it only appears in unison with the word "*San*". KN smoothing generalized this by not setting the unigram probability to be proportional to the number of occurrences of a word, but instead to the number of different words that it follows.

3.2 BERT

Bidirectional Encoder Representations from Transformers [BERT] is one of the latest milestones in natural language processing, with accuracies far superior to previous language models in its handling of language-based tasks. The model was built and published by Jacob Devlin and colleagues of his from Google in 2018, and since then the model have been made open-sourced and made available to download pre-trained versions on data from the BooksCorpus and the English Wikipedia (Wikipedia.org, 2022) totalling 16 GB of Data and 3.3 billion words. The classic BERT-model is built on multiple transformer blocks. The transformers are deep learning models that handles sequential data, similar to recurrent neural networks. However, the transformers do not need to process organized sentences like the RNN does, to understand the context of the sentences. The result of this is that transformer blocks can work parallel on complete sentences in order to understand the context of it which reduces run-time on training significantly. BERT-Base refers to a BERT-model that uses 12 transformers, and BERT-Large uses 24 blocks. Just like the transformers, BERT inputs the sequential data and runs it through each transformer with help of a feed-forward network. It then outputs a vector that is sent through a classification-layer and finally decoded into a sentence again (Alammar, 2018).

BERT is trained with a masked language modelling task and a next sentence prediction task. The masked modelling task replaces approximately 15% of the input tokens with a mask token, then the model has to predict these masked tokens. The next sentence prediction task provides the model with two sentences and the model has to predict whether or not the second sentence appears after the first sentence.

3.3 RoBERTa

Since the introduction of BERT, many have tried to improve the model to make it even greater. In 2019, Facebook introduced the "*Robust Optimized BERT approach"* [RoBERTa] as an improved version of the BERT by removing the next-sentence predictor, training the model on larger samples with bigger batches, and changing the masking (Yinhan Liu, 2019). Their belief was that BERT was under-trained and under-optimized and wanted the rectify this by increasing the training size. In total, RoBERTa uses 100 times the size of the training data than BERT with the training size totalling 160 GB of data. This have led to increased performances, but at the cost of 4-5 times longer training time than with BERT (with the Large model). Because of these changes to the model, in theory, RoBERTa should be able to outperform the classical BERT-model when predicting the the missing word in the Sentence Completion Challenge.

4.0 Implementation

4.1 N-grams

The n-gram language model that was used in this analysis was first created at the University of Sussex for the Advanced Natural Language module, then the author revised the model to include a trigram model as well as the pre-existing unigram and bigram. To generate answers to the multiple-choice questions, the n-gram models predicts the highest probable words from each alternative.

4.1.1 Hyper-parameter settings

The n-gram model was tuned by using nested for-loops to increase the size of the training data, Out-of-Vocabulary threshold and the Discount for smoothing, testing the model on the MRSCC. This helped to illustrate how the accuracy changed whenever a hyper-parameter was changed in respect to other parameters so that the relationships between each parameter also could be explored in the process. Appendix 1 below shows the tuned hyper-parameters and how the parameters affected the accuracy of the model.

Table 1

Model		Specifications	Accuracy
Ngram	Unigram	2 OOV words 0.75 Discount	24.904 %
	Bigram	2 OOV words 1 Discount	25.288 %
	Trigram	2 OOV words 0.75 Discount	32.308 %
BERT	BERT	Large Uncased Base Only No pooling	78.846 %
	RoBERTa	Large Base Only No pooling	76.731 %

4.2 BERT and RoBERTa

The BERT-models are pre-trained on 16 GB of masked language texts from Wikipedia and the BooksCorpus and RoBERTa is pretrained on 160 GB of masked language texts. These pre-trained models were acquired through the Hugging Face library.

4.2.1 Hyper-parameter settings

BERT and RoBERTa was tuned on four hyper-parameters displayed in table 2:

Table 2

Hyper-parameters	Specifications
N Transformer blocks	Base uses 12 transformers, Large uses 24 transformers
Case sensitivity	Whether or not the model is sensitive to upper- and lower-cases (Cased vs. Uncased)
Result Method	With or without pooling
Pooling method	Base only, min, max, mean

The pooling method mentioned in the table above have four different alternatives: Base, minimum maximum and mean. With Base the model only considers the first token (the base) of every option word and choose the token that has the highest probability of fitting into the place. With max, min and mean, the model takes the selected option, e.g., the mean of the embedding of every token of the option words as the option embedding. It then chooses the option which is the most similar to the word that has the highest probability of fitting into the sentence.

5. Evaluation

When making random guesses to predict the correct answer from a multiple-choice task with

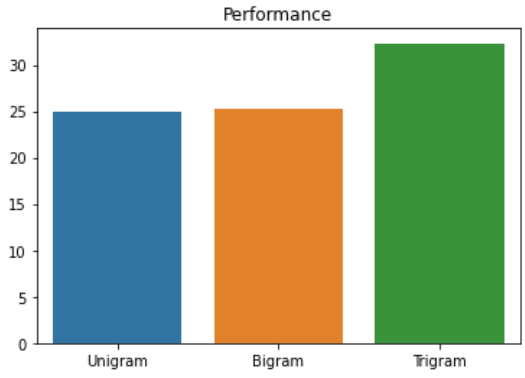
five alternatives, such as in MRSCC, the probability to get a correct answer lay at 20 % (= 1/5). Taking this into account when evaluating the n-gram models it is clear to see that they only perform slightly better than random predictions, with the unigram and bigram performing at approximately 25 % accuracy, and the trigram being the best performer with an output of 32 % accuracy. From the research paper by Geoffry Zweig and Christopher Burgees (Geoffry Zweig, 2011) we learn that with a smoothed trigram, like the one used in this research, they received results of 36 % accuracy when training and testing the model on the same data as in this research.

Table 3

	Unigram	Bigram	Trigram
Entropy	1.035	1.815	0.967
Perplexity	731.507	139.927	47.244
Accuracy	24.904 %	25.288 %	32.308 %

An interesting observation when looking at the results of the unigram and bigram models in table 3 and figure 1 is that despite the bigram only slightly outperforms the unigram by 0.3 percent accuracy, the perplexity of the models show that the bigram is capable of predicting words with a higher confidence.

Figure 1



(This figure is a barplot of the performances of the n-gram model)

Interestingly, table 1 shows that BERT outperforms RoBERTa by 2 %. This could be because of the ambiguity of the alternatives to the missing word and because of the evolution of writing in the 100 years since Sir Arthur Conan Doyle wrote his Sherlock Holmes novels. By analysing the predicted answers from BERT and RoBERTa we see that the models give the same answer 70 per cent of the time. 15 per cent of the predictions are answered correctly by one, but not the other, and 15 per cent of the predictions are answered incorrectly by both models. This means that 85 per cent of the predictions are answered similarly by both models. When looking at the questions that both models predicted wrongly, it is clear to see that the masked words could very likely be one of many of the alternatives, so the answers are more ambiguous than others. E.g., One question both models got wrong was this:

*'There is no <**MASK**> between them, but they all open out into the same corridor.'*

A) 'understanding'
B) 'communication'
C) 'difference'
D) 'intrigue'
E) 'issue'

In this question the answer could easily be anyone of the alternatives, but both models chose *'difference'*, when the answer was *'communication'*.

6.0 Conclusions

The Sentence Completion Challenge is proving itself still to be challenging today as it has been since its introduction in 2011, however, advancements have also clearly been made since then. Zweig and Burges used a Latent Semantic Analysis as their best performing model with a score of 49 %. With use of the state-of-the-art pre-trained BERT and RoBERTa models, the accuracy of correctness is improved to 75-80 per cent. The remaining errors could be chalked up to their ambiguity in the alternatives. From the performances of the n-gram models it is possible to see that having a better understanding of the sequences before the

predicted words will increase the accuracy of the model. BERT takes this even further with the ability to understand the entire sentence (apart from the masked word) before making its predictions.

6.1 Further work

The n-gram model was the baseline model of the two proposed models in this paper. In this research the word-sequences were calculated by using the chain rule for probabilities as shown in the equation in section 3.1. When selecting the alternative with the n-grams solution is to always pick the highest probable of the alternatives, however, introducing a form of randomness by using a Boltzmann sampler, maybe accuracy could be increased. To use the Boltzmann sampler, however, another hyper-parameter would need to be introduced, namely temperature. To illustrate, to transform probability distributions into Boltzmann distributions, the following formula is used:

$$q_i = \frac{\exp\left[\ln(p_i)\ /T\right]}{\sum_i^N \exp\left[\ln(p_i)\ /T\right]}$$

Where N is the number of alternatives and T is a parameter. In physics this distribution is the one of a system at equilibrium with temperature T. If T = 1, the probability distribution p = q, i.e. it reverts back to the probability distribution. This could also be used in the BERT and RoBERTa Language Models when predicting the most accurate alternative with the respective language models.

The BERT and RoBERTa both get very accurate results with their pre-trained models, however, run-time is quite taxing and could be something to improve upon. It would be interesting to see if DistilBERT, with its faster run-time, could achive results close to aforementioned models.

References

Alammar, J., 2018. *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning).* [Online]
Available at: https://jalammar.github.io/illustrated-bert/
[Accessed 02 April 2022].

Geoffry Zweig, C. J. B., 2011. The Microsoft Research Sentence Completion Challenge. 20 Febuary.

Horev, R., 2018. *TowardDataScience.com.* [Online]
Available at: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270
[Accessed 23 April 2022].

Stanley F. Chen, J. G., 1999. An empirical study of smoothing techniques for language modeling. 28 January.

Wikipedia.org, 2022. *www.Wikipedia.org.* [Online]
Available at:
https://en.wikipedia.org/wiki/BERT_(language_model)#:~:text=BERT%20was%20created%20and%20published,almost%20every%20English%2Dlanguage%20query.

Yinhan Liu, M. O. N. G., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *https://arxiv.org/pdf/1907.11692.pdf*, 26 July.

Appendices

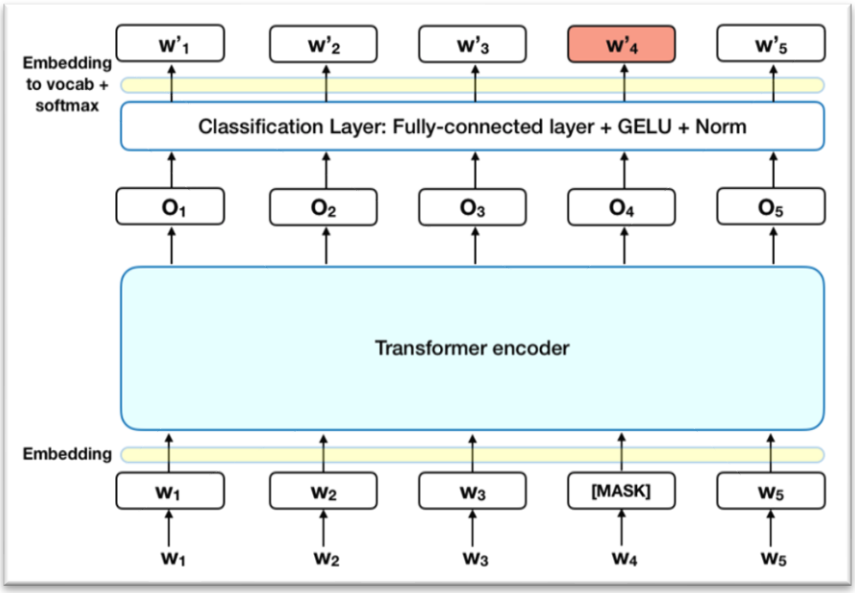
Appendix 1

Number of files	OOV threshold	Discount	Unigram Accuracy	Bigram Accuracy	Trigram Accuracy
10	2	0,5	20,865 %	19,808 %	21,442 %
10	2	0,75	20,962 %	19,327 %	20,673 %
10	2	1	21,154 %	19,423 %	20,192 %
10	5	0,5	17,308 %	18,077 %	22,212 %
10	5	0,75	15,865 %	18,558 %	21,731 %
10	5	1	16,635 %	19,135 %	19,712 %
10	10	0,5	17,404 %	21,058 %	21,923 %
10	10	0,75	15,769 %	19,904 %	21,923 %
10	10	1	14,423 %	18,750 %	17,308 %
50	2	0,5	24,135 %	20,000 %	25,865 %
50	2	0,75	24,231 %	20,385 %	24,615 %
50	2	1	24,135 %	21,346 %	23,462 %
50	5	0,5	23,462 %	18,750 %	21,731 %
50	5	0,75	23,365 %	18,846 %	21,154 %
50	5	1	23,269 %	19,904 %	20,288 %
50	10	0,5	22,596 %	19,327 %	20,192 %
50	10	0,75	22,404 %	19,712 %	20,096 %
50	10	1	22,885 %	20,096 %	19,231 %
100	2	0,5	24,135 %	22,404 %	27,404 %
100	2	0,75	24,038 %	22,212 %	27,692 %
100	2	1	24,038 %	22,981 %	27,115 %
100	5	0,5	24,135 %	21,827 %	25,769 %
100	5	0,75	24,038 %	21,923 %	24,615 %
100	5	1	24,135 %	21,442 %	23,077 %
100	10	0,5	22,788 %	20,192 %	23,462 %
100	10	0,75	23,365 %	20,481 %	22,308 %
100	10	1	23,269 %	20,962 %	22,019 %

Appendix 2

Model Checkpoint	Result Method	Pooling Method	Accuracy
bert-base-uncased	base_only	None	75,000 %
bert-base-uncased	all	max	49,038 %
bert-base-uncased	all	min	45,192 %
bert-base-uncased	all	mean	47,500 %
bert-base-cased	base_only	None	70,192 %
bert-base-cased	all	max	52,692 %
bert-base-cased	all	min	54,904 %
bert-base-cased	all	mean	55,000 %
bert-large-uncased	base_only	None	78,846 %
bert-large-uncased	all	max	53,942 %
bert-large-uncased	all	min	52,019 %
bert-large-uncased	all	mean	55,000 %
bert-large-cased	base_only	None	76,250 %
bert-large-cased	all	max	56,058 %
bert-large-cased	all	min	57,788 %
bert-large-cased	all	mean	58,846 %
roberta-base	base_only	None	71,346 %
roberta-base	all	max	54,712 %
roberta-base	all	min	56,442 %
roberta-base	all	mean	56,346 %
roberta-large	base_only	None	76,731 %
roberta-large	all	max	55,288 %
roberta-large	all	min	55,481 %
roberta-large	all	mean	57,596 %

Appendix 3



(Horev, 2018) An illustration of a BERT Masked-Language Model