Advanced NLP Report, The Microsoft Research Sentence Completion Challenge

Abstract

The following paper tackles the Microsoft Research Sentence Completion challenge, which was researched by Geoffry Zweig and Christopher J.C. Burges back in 2011. Different language models and approaches will be discussed, evaluated, and compared to conclude which models perform better at the word prediction task.

Keywords: NLE, Microsoft Research Completion Challenge, language modelling, modelling semantics, word prediction.

1. Introduction

Natural language engineering (NLE) has gained much popularity in the field of Artificial Intelligence (AI) and Machine learning (ML) in the past decade as researchers and developers work on creating new approaches and optimising existing ones to solve complex problems in this field. Semantic analysis, predictive text, language translation, e-mail filters, text-to-speech, and many more are typical applications of NLE. This paper will explain how semantic modelling of text is used to carry out a word prediction task. Word prediction is essential in the field of NLE as it aims to improve applications like speech recognition, machine translation, speech recognition, and many other applications. Being able to create a reliable system to predict words is vital, as it works on refining the keystroke saving rate (KSR), which is the number of keystrokes that are predicted correctly by the system; better systems tend to save keystrokes on behalf of the user; thus, better performance is achieved when the number of KSR is higher; as a result, less time and effort is needed for the user to carry out a query, such applications are being used by Google's search engine to predict a query, Amazon's Alexa, and many more applications currently used day-to-day by multiple users (Hamarashid, Saeed and Rashid, 2022, p. 3). The systems will be modelled using two different approaches; the first one will be training N-gram models (unigram, bigram and trigram) on 1040 sentences selected from five of Sir Arthur Conan Doyle's Sherlock Holmes novels which were collected from the project Gutenberg dataset, for each of the 1040 sentences there are four imposter sentences, where an impostor word with similar occurrence statistics has replaced a single fixed word in the original sentence, the impostor words were suggested by a language model trained on over 500 19th century novels, accordingly, human judges picked the best four impostor words (Zweig and Burges, 2011, p. 1). The second model is a simple neural network (NN) Word2Vec model, which Tomas Mikolov proposed(Mikolov *et al.*, 2013). This model will be trained separately using Google's pre-trained embeddings and Holmes embeddings. Both models will be explained and evaluated based on the predicted word out of the five available choices.

2. Models and Approach

2.1. N-grams (Unigram, Bigram and Trigram)

Language models are models that assign probabilities to sequences of words. Unigrams, bigrams, and trigrams are examples of a language model; these models can assign a probability to an upcoming word by computing the probability of a sentence of a sequence of words. Here, the trigram model is compared with the unigram and bigram models as benchmarks, so the main focus is on explaining how the trigram works and how well it performs on the dataset. The trigram model considers only three words at a time, the target word, and the previous/ next n-1 words (i.e., 3-1=2), by creating a context window.

- The probabilities can be found by applying the chain rule of probabilities. The chain rule is used to convey the relationship between the joint probability of a sequence of words and the conditional probability of a word given preceding words.

$$P(W1, W2, W3, W4, \ldots\ldots, Wn) = \prod_{i=1}^{n} P(Wi|Wi-(n-1), \ldots\ldots Wi-1)$$

- Markov's assumption can be applied to the chain rule, which states that the probability of an upcoming individual word can be computed by looking at the previous n-1 words and without considering the whole history of preceding words; therefore, for the trigram case, the previous two words will be taken into consideration. After the approximation of each word in the product is found, an estimate can be deduced by using maximum likelihood estimation (MLE) on the whole corpus, which is done by getting counts of a certain trigram from the corpus and normalising those counts by the same trigrams that share the same first word, so now they range between 1 and 0.

$$P(Wi|Wi-(n-1), \ldots., Wi-1) = \frac{freq(Wi(n-1), \ldots., Wi-1, Wi)}{freq(Wi-(n-1), \ldots., Wi-1)}$$

- Some words in the training data might not occur in the testing data, and the model will assign zero probability to those; consequently, perplexity will not be calculatable; therefore, a pseudoword "__UNK" is added if the word is not seen in the testing data, then the probability of this unknown word is estimated from its counts just like any other word in the training set.
- Another issue which needs to be addressed is the start- and- end of a sentence; this is solved by also adding pseudowords "__START" and ["__END","__END"] to each n-gram sentence for the trigram.
- A common practice for n-grams is to represent the probabilities as log probabilities to minimise numerical underflows when multiplying probabilities together.
- Perplexity (PP) is a typical evaluation metric to evaluate language models, which is the inverse of the probability of the test set normalised by the number of words.
- Finally, smoothing is done to account for n-gram combinations that occur in the training set but do not occur in the testing set; this will be done using Kneser-Ney (KN) smoothing with absolute discounting; this will take off some probability mass from more frequently

occurring events and assign that to events that never occurred or are less frequent ((Jurafsky and Martin., 2021, pp. 2–8)

2.2. Word Similarity using Word2Vec Model

Word2Vec is a common method of generating word embeddings to represent words in a vector form, where each word has a unique vector representation, as machines cannot process the raw string format of words. However, they can do that for the vector format. With the use of word encoding, each word vector will be mapped into a particular category, then Word2Vec will group vectors of similar words based on their appearances in the context of a text after being trained using the words' embeddings and, and in this case, the Word2Vec model will be trained on the Google News and Holmes words' embeddings. The Word2Vec model gained popularity in the field of NLP because of its architecture which is made up of the following two neural network models:

- Continuous Bag of Words Model (CBOW): This model tries to predict a word given the context; this is done by using the continuously distributed representation of the words in the context and attempting to predict the target word. The structure of this model can be seen in Figure 1 below.

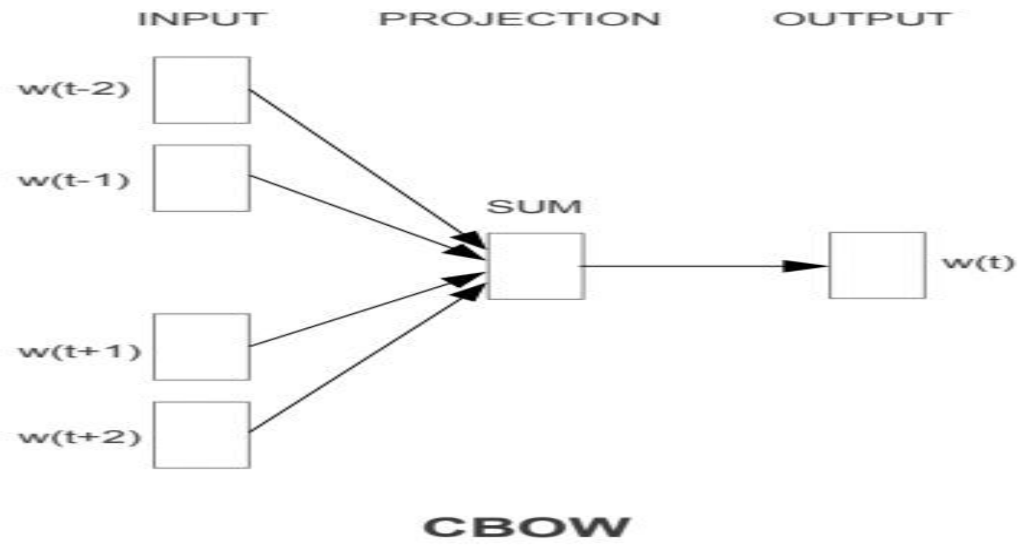

Figure 1: CBOW architecture(Mikolov *et al.*, 2013, p. 5)

- Continuous Skip-Gram Model: This model is a form of a neural network with one hidden layer, the functionality of this model is to predict the previous and the next word given a current word as an input vector to a log-linear classifier with a continuous projection layer, it can be thought of as the opposite of CBOW. The learning process continues and is optimised with each run until an optimal output is found (Vatsal P., 2021). The structure of this model can be seen in Figure 2.

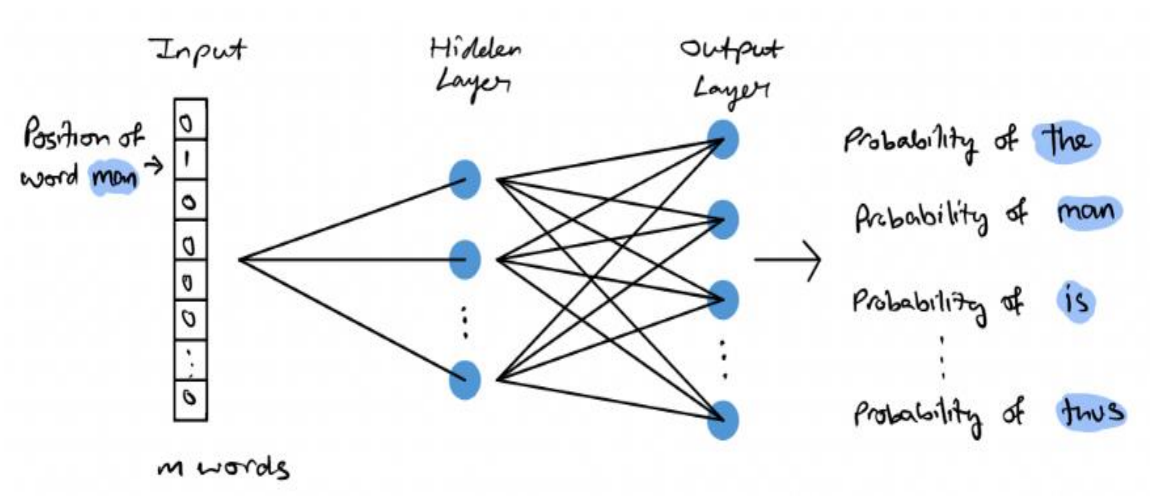

Figure 2: Continuous Skip-Gram architecture(P., 2021)

3. Evaluation and Hyper-parameters

Now that the models have been explained, the next step is training the models on the Holmes dataset and experimenting with different hyper-parameters to see which combination yields the best outputs on the 1040 questions.

3.1. Evaluation of N-grams Language model

- Perplexity: The first evaluation metric of the models' performance is perplexity; the model was trained on 100 files from the training data and tested on 100 files from the testing data; the resulting perplexity of the three N-grams on these files can be seen in figure 3 below

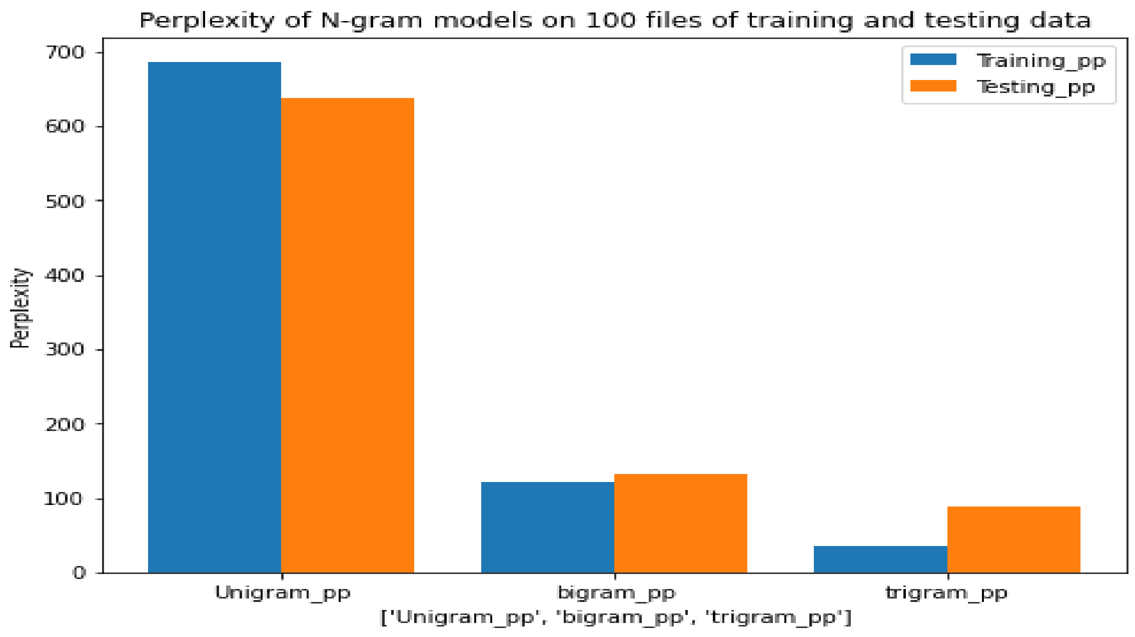

Figure 3: Perplexity of the three N-gram models on the training and testing data

From the Perplexity bar plot, it can be noticed that as N is increased for the N-gram language models, the perplexity decreases, meaning that the trigram gives more information about the word sequences compared to the other two N-grams; therefore, the perplexity achieved for the tri-gram is less as expected.

- Accuracy: The models will be evaluated on the 1040 questions' sentences mentioned, where the models will try and predict a word given a context. A question and a SCC_reader class were developed in the labs; the question class reads in the question files and preprocesses them for each N-gram, while the SCC_reader class evaluates each N-gram, given the correct answers for each question, so the evaluation metric used here will be accuracy. Moreover, hyper-parameter tuning was conducted. It should be noted that the models were trained on 100 files out of the 261 training files due to computational complexity requiring much time to run the training process on all files. The tri-gram performed best with a word frequency threshold of K=2 and a number of files=100; it should be noted that N-gram language models perform better when fed more significant amounts of data(Jurafsky and Martin., 2021, p. 8). The performance of each model can be seen in figure 4 below.

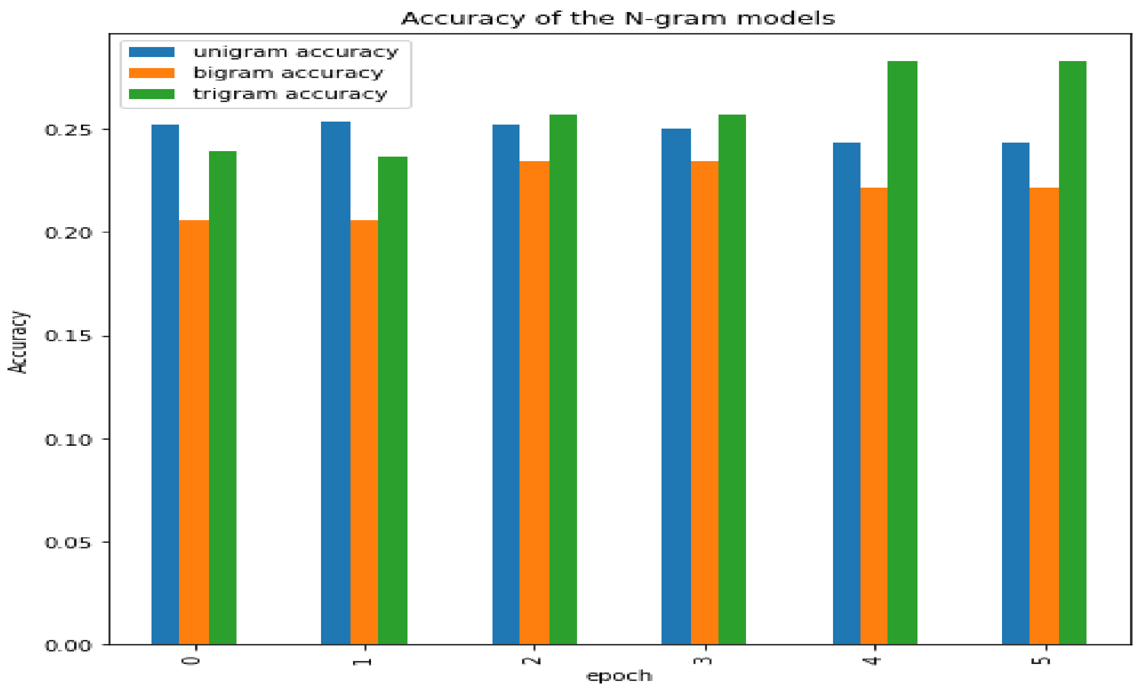

Figure 4: Accuracy of N-gram models on question files

From this bar plot, it can be noticed that there is a rising trend in the accuracy of the models as the number of training files (20,60,100) fed to the models is increased; this supports the findings suggested by researchers in this field(Jurafsky and Martin., 2021, p. 8), this is also the reason why the uni-gram is outperforming the bigram as N-grams perform better when they are fed with larger amounts of training data, another reason for these accuracy results, is that there might be some unseen bigram combinations in the training data.

3.2 Evaluation of Word2Vec Model

Two Word2Vec models were generated, the first one was using preprocessed Holmes training data; after experimentation, the best results were obtained using the following model parameters:

- Vector size= 300, which denotes the words' vectors' dimensionality.
- CBOW method, by setting skip-gram (sg=0).
- Context window=5, is the maximum distance between the current and predicted word in the context.
- Negative=5, which is using negative sampling to indicate the number of noise words to be drawn. The negative sampling objective is to maximise the similarity of words in the same context and minimise it when they appear in different contexts, so negative sampling only updates the necessary weights after each epoch(Karimi, 2021).
- Min_count=5, set as a threshold to ignore words that have a frequency less than 5

The second Word2Vec model was trained on Google news pre-trained embeddings with vector sizes 300 and 3 million words which can be found here (Google, 2013). The results obtained support the findings of Mikolov and his research colleagues, as Word2Vec was much more computationally efficient than using N-grams. In addition, much higher accuracy was obtained, keeping in mind that the N-grams were trained only on 100 files, while the Holmes Word2Vec model was trained on all the training data, i.e. 261 files. The genism library(Řehůřek, 2019) was used to implement the Word2Vec models; this library allows the user to tune the parameters of the model by using the preferred architecture, as expected, by using cosine similarity/avg similarity to calculate the distance between a word from the five choices and the questions in lying in the vector space(Mikolov *et al.*, 2013, p. 4), it was found that training the Holmes Word2Vec model with CBOW yielded better accuracies on the 1040 questions, compared to the skip-gram, as the objective of the model is to predict a word given a context, which is what a CBOW is intended to do, figure 5 shows the accuracies of both models.

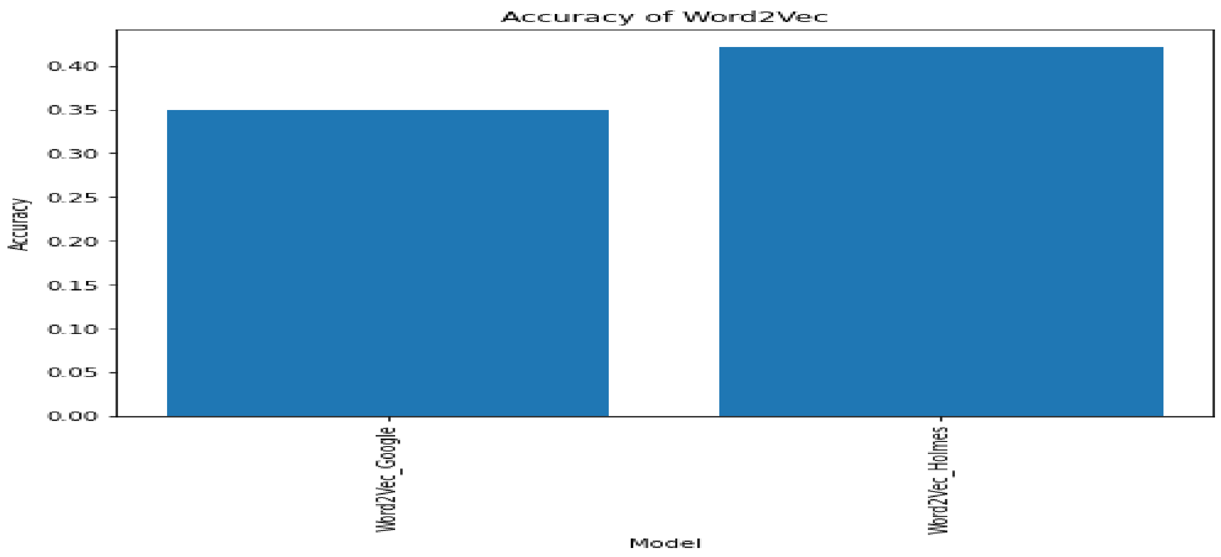

Figure 5: Accuracy of Word2Vec Models

3.3 Word2Vec VS N-grams and Error Analysis

After explaining the models, explaining their underlying architecture, and evaluating them separately, it is convenient to compare their performance, an overall comparison of all the models' accuracies on the 1040 questions after hyperparameter tuning was conducted. Furthermore, error analysis will be discussed, and the results of the models' accuracies can be seen in figure 6.

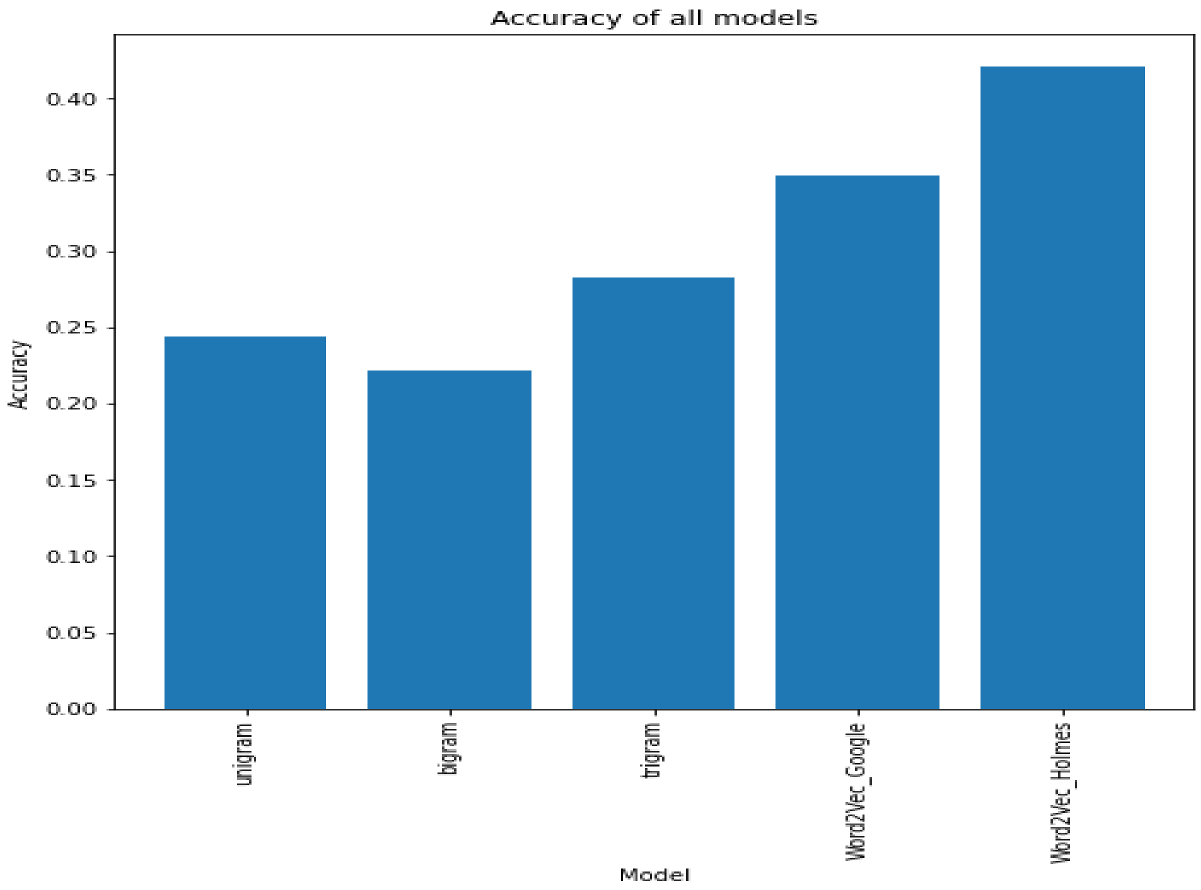

Figure 6: Accuracy of all models

It can be noticed that the Word2Vec models are outperforming the N-gram language models, and once again, this is due to their underlying architecture, which behaves as a NN. NNs are used in many fields in NLE as they can mimic the way a human brain works; they can be trained on massive datasets and optimised to produce state of the art applications in speech recognition, machine translation, and other applications. Another reason for Word2Vec scoring higher accuracies is the amount of data it was trained on; as mentioned before, the N-grams were only trained on 100 files of the Holmes data, while the Word2Vec models were separately trained on the whole Google news data, and the Holmes training data, thanks to Mikolov's research findings on how to train simple NN models on large data sets while reducing computational complexity. Another approach to evaluate the models rather than accuracy is the amount of shared correct answers and the amount of shared wrong answers; this can be seen in figure 7.

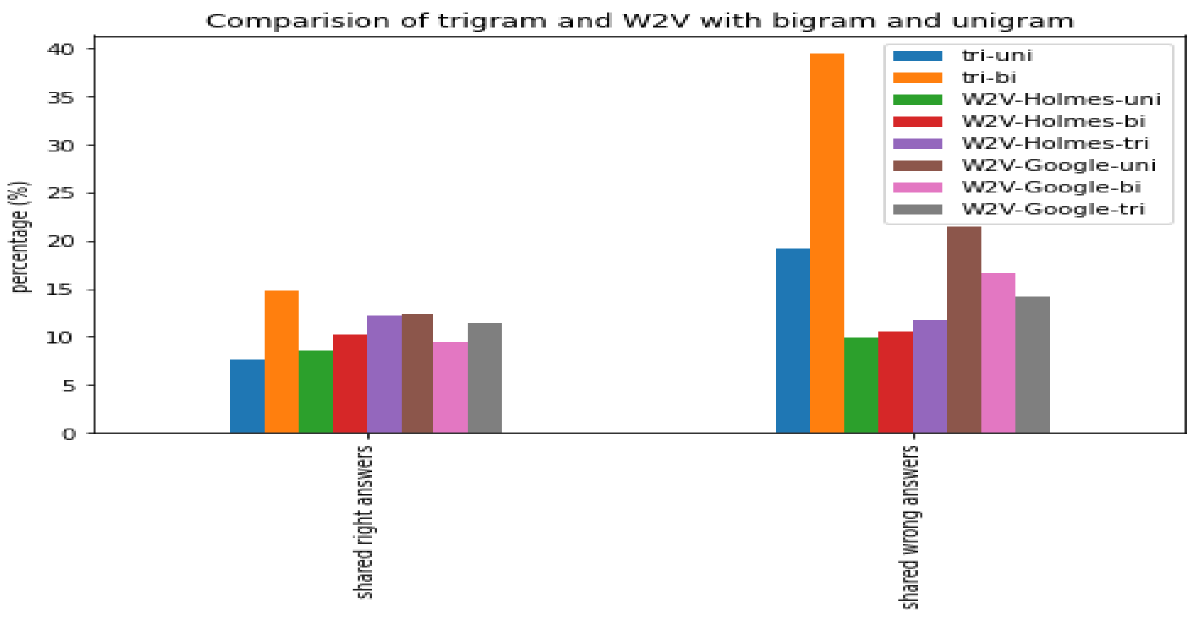

Figure 7: Amount of shared right/wrong answers of all models.

From figure 7, which can be used as an error analysis method, it can be noticed that tri-grams and bi-grams have the highest share of wrong and right answers, while Google's Word2Vec share the highest right answers with the uni-gram. On the other hand, the Holmes Word2Vec model shares the highest number of correct answers with the trigram. This solidifies the findings that Holmes Word2Vec and the tri-gram are the best out of the two approaches in predicting the correct answers for the questions.

4. Conclusion and Further Work

The main task was to predict a word given a question sentence, and from the proposed models, the Holmes Word2Vec model had the highest performance, followed by Google's Word2Vec and the tri-gram, while the uni-gram and bi-gram acted as a benchmark. The results were as expected, as Mikolov's Word2Vec model is more efficient and can give better results due to the learned word embeddings which carry semantic information of the words in the vector space. Improvements can yet be made using multiple workers (parallel computation). Increasing the training size and vector size would increase the quality of word embeddings up to a convergence point for Word2Vec (Mikolov *et al.*, 2013, p. 10). For evaluation of N-grams, another metric can be used, which is entropy; entropy is a measure of information, where entropy rate and cross-entropy can be used alongside perplexity to evaluate a language model(Jurafsky and Martin., 2021, pp. 21–24). Using more sophisticated state of the art models like BERT, GPT, and XLNet is also an option. However, they require much more computational power and have longer training processes(Arunachalam, 2021), which is the main reason why Mikolov introduced Word2Vec in the first place. Finally, text prediction is a task under continuous research in the field of NLE. While the NN and deep learning technologies are evolving exponentially, more modern architectures with better performance will be seen in future practical applications.

References

1. Arunachalam, A. (2021) ‘An illustration of next word prediction with state-of-the-art network architectures like BERT, GPT, and XLNet’. Available at: https://medium.com/mlearning-ai/an-illustration-of-next-word-prediction-with-state-of-the-art-network-architectures-like-bert-gpt-c0af02921f17.

2. Google (2013) *Google Code Archive*. Google. Available at: https://code.google.com/archive/p/word2vec/.

3. Hamarashid, H., Saeed and Rashid, T. (2022) ‘A comprehensive review and evaluation on text predictive and entertainment systems’. Available at: https://arxiv.org/ftp/arxiv/papers/2201/2201.10623.pdf#:~:text=The%20main%20aim%20of%20word,number%20of%20KSR%20is%20high.

4. Jurafsky, D. and Martin., J. (2021) ‘N-gram Language Models’, in *Speech and Language Processing*. Available at: https://web.stanford.edu/~jurafsky/slp3/3.pdf.

5. Karimi, A. (2021) ‘NLP’s word2vec: Negative Sampling Explained’. Available at: https://www.baeldung.com/cs/nlps-word2vec-negative-sampling.

6. Mikolov, T. *et al.* (2013) ‘Efficient Estimation of Word Representations in Vector Space’. Available at: https://arxiv.org/pdf/1301.3781.pdf.

7. P., V. (2021) ‘Word2Vec Explained’. Available at: https://towardsdatascience.com/word2vec-explained-49c52b4ccb71.

8. Řehůřek, R. (2019) *models.word2vec – Word2vec embeddings*. Gensim. Available at: https://radimrehurek.com/gensim_3.8.3/models/word2vec.html.

9. Weeds, J. (2021a) ‘Language Modelling’.

10. Weeds, J. (2021b) ‘Lexical and Distributional Semantics 2’.

11. Zweig, G. and Burges, C. (2011) ‘The Microsoft Research Sentence Completion Challenge’.