**COMP40020 Assignment 2**

**17350796**

In this report word similarities with word embedding algorithms and n-gram language models are explored. The corpus that is used is all the titles, title bodies, and comments from the top 20 posts from r/ireland subreddit.

**Comparing 10 words from the generated corpus to a pre-trained language model**

**Text

Description automatically generated**As you can see the ten words that were selected are in the array in the image below. The method returns the word and the similarity score to the query word. The similarity score is generated from a Cosine similarity calculation of the word vector values.

This Word2vec model is a pretrained model. The original was trained on 100 billion words obtained from the Google news dataset. Although, in this version a pruned iteration is implemented. Only 43981 words are used. Other popular pre-trained models are the Glove model and Fasttrack. There use different algorithms to learn word associations. For example, Fasttrack uses character n-grams in their model. This is beneficial as it means it can handle words not in its vocabulary which word2vec cannot. Fasttrack can take into account sub words. It is also more effective at recognising a phrase that is expressed as a single word. Thus, it can depend on the language for which algorithm is better at learning the vector representations.

**Building a language model from the corpus and exploring generated sentences**

Graphical user interface, text

Description automatically generatedIn this section a language model was built using the reddit text. Due to data coming from reddit it had to be cleaned. The below function was used to remove the emojis from the text:

Links and extra spaces were also removed. The python regex module is very useful for this:

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generatedA language model using unigrams, bigrams, trigrams, and quadgrams were all made.

A series of sentences with different lengths was generated. As you can see from the below image the sentences do not make sense.

Text

Description automatically generated

In order to clean the sentences the following code was used:

Graphical user interface, text, application

Description automatically generated

This code was taken from (Tan, 2019). This function ensures that the sentence starts with the start padding token and ends with the end padding token. The detokenize method ensures the returned string has correct spacing and punctuation. This provides cleaner results:

Graphical user interface, text, application, email

Description automatically generated

Some of the sentences are quite comical and very much in line with r/ireland discourse. As is shown the sentences get more coherent as the number of n-grams is increased. The trigram and four-gram sentences make more sense than the unigram sentences. This is due to being able to take into account more context. More probabilities of different words occurring together or relationships between words can be captured with higher numbers of n-grams. For example, a unigram model only considers one-word sequences while the four-gram considers four-word sequences. The unigram model assumes each word is independent. It is also shown that shorter sentences provide more coherent results. Again, this due to the having to consider more relationships between different words. A higher number n-grams would potentially allow for longer sentences. However, the data size is limited meaning some n-grams may not appear in the data frequently.

**A language model with words from the corpus that begin with ‘B’**

A model for words was built using only words that begin with the letter ‘B’. For this the text had to be processed in a different manner as the goal was to generate words as opposed to sentences. This is more similar to the Fasttrack algorithm. The code below shows filtering the text for words beginning with ‘b’.

A picture containing text

Description automatically generated

Graphical user interface, text, application, email

Description automatically generatedThe word tokenizer from NLTK was used as opposed to the sentence tokenizer:

The total number of tokens obtained was 2938. Here are some of the results of the model:

A picture containing scatter chart

Description automatically generated

Text

Description automatically generatedAs you can see the results fare better with more n-grams (the number of n-grams increases down the arrays). Words like ‘ban’, ‘ring’, and ‘brush’ are being generated. The same function was used from the other model to ensure each word starts with the padding character and ends with the padding character:

This generates clearer results. Again, the unigram model does not provide any actual words. It assumes that probability of each letter occurring is only based on its frequency in the training data.

Scatter chart

Description automatically generated with medium confidenceText

Description automatically generatedIt works well when given a text seed of ‘b’ as this means it is the starting point of the letter sequences:

**Limitations of n-gram models and areas for improvement**

There are number of limitations associated with n-gram models. N-gram models cannot take into account the relationship between words that are far a part in an n-gram. This can ignore important context in certain cases. This can cause inaccurate predictions.

An issue seen with the models used in this report was the quality of the training data. The corpus was made up of text scraped from social media. This meant it likely contained large amounts of slang, abbreviations, misspellings, grammatical issues, and colloquialisms. Pre processing techniques like removal of punctuation, tokenisation, detokenization, and emoji removal improved the consistency and quality of the data.

Model evaluation can also be challenging. A common evaluation technique is perplexity. This is considered an intrinsic evaluation technique. This means it is an evaluation of the model itself, not an evaluation of the model in a specific task. Perplexity is the inverse probability of the test data set, normalised by the amount of words (Campagnola, 2020). Lower perplexity is indicative of being more certain of a prediction. Perplexity is sensitive to size, a small dataset can result in low perplexity scores (Serge, 2021).

As Phillip et al (2006) describes that n-gram models struggle to reprsent extended phrase histories. Meaning that a model with a large amount of n-grams has to capture all possible n-word sentences. This can be computationaly expensive and requre a lot of storage. Phillip et al (2006) suggests a reduce n-grams’ approach. This approach excludes infrequent n-grams from the model.

N-gram models as seen from the examples in the notebook depend on the data on which it is trained. The sentences generated were reflective of the ireland subreddit. The model would not generalise to other use cases. Even by increasing the training data, it is not possible to account for all possible n-grams. Thus, smoothing techniques were introduced to increase generalisability of models. These techniques “smooth” out probability scores. For example, Kneser-Ney smoothing is a technique whereby probability estimates for n-grams is discounted by a particular amount and redistributes it to n-grams that are unseen. This technique has been shown to be more effective than other smoothing techniques like Laplace.

**Conclusion**

I have gained a better understanding of how language models are built and operate. Moreover, it was interesting how n-gram models compare to word embedding algorithm like Word2Vec. Unlike language models, Word2Vec does not predict sequences of words. It captures the relationships between words in a corpus. While n-gram models have clear limitations, they were one of the earliest concepts of natural language processing and are still very much in use today. Interestingly, word embedding algorithm fasttrack uses letter n-grams in its algorithm. This is common in modern day applications, n-gram models are used in conjunction with other algorithms. N-gram model are used for applications like speech recognition, spelling correction, and machine translation. While there are many limitations to n-gram models