**COMP40020 Assignment 2**

**17350796**

In this report word similarities, n-grams, 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. 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.

**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 process in a different manner as the goal is to generate words. 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. Same function is used from the other model to ensure each word starts with the padding and ends with the padding:

This generates clearer results. Again, unigram model does not provide any actual words.

Scatter chart

Description automatically generated with medium confidenceIt 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**

**Conclusion**

I have gained a better understanding of how language models are built and operate. 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. Interestingly, word embedding algorithm fasttrack uses letter n-grams in its algorithm. I can see the importance of n-gram model for applications like speech recognition, spelling correction, and machine translation. While there are many limitations to n-gram models