NLP Assignment 3

- *Shreyash Arya (2015097)*

→ The data corpus for the comp.graphics and rec.motocyles is created by combining all the .txt files as mentioned in the assignment doc.
→ The average sentence length is considered 14 for both the classes by experimenting with the average number of words per sentence.
→ Log probabilities have been taken.
→ Add-one smoothing is used everywhere where required.
→ Models have been saved for all the unigrams, bigrams and trigrams for both the corpora.

Preprocessing:

- Headers and footers (first and last parts separated by '\n') are ignored while combining the .txt files into the single corpus.
- The whole corpus has been preprocessed by removing the punctuations, HTML tags, ASCII characters, lowering case all the words, word contractions and lower-casing all the words.
- Stopwords are removed for only the generation of unigram sentences otherwise the sentence would mostly consist of stopwords because of their high frequency of occurrence in corpus.

Part 1: Generative Model

1. Generate 1 sentence using unigram model

The sentence has been generated by considering the words with the highest frequency count.

2. Generate two different sentences using bigram model.

Sentences have been generated by considering the first bigram with the highest probability.

For the starting word of the bigram, the word which occurs most frequently at the start of the sentences has been considered from each of the corpora.

For tackling the problem of repeated bigrams if we have the same word followed repeatedly or the bigrams with equal probability, the log probabilities have been multiplied by 2 to reduce it's preference while selecting the bigrams in the next iteration.

For two different sentence generation, the most frequent word at the start of the sentence and the second most frequent word at the start of the sentence is considered.

Regex(r‘[.!?]+[>)\'}\]\"]*(\s|\n)+[\'\"{\[\(<]*[A-Za-z0-9]+’) is used for finding the sentence boundaries and words at the start of the sentence.

3. Generate two different sentences using trigram model.

Sentences have been generated by considering the first trigram with the highest probability.

For the starting word of the trigram, the word which occurs most frequently at the start of the sentences has been considered from each of the corpora.

For tackling the problem of repeated trigrams if we have the same word followed repeatedly or the bigrams with equal probability, the log probabilities have been multiplied by 2 to reduce it’s preference while selecting the trigrams in the next iteration.

For two different sentence generation, the most frequent word at the start of the sentence and the second most frequent word at the start of the sentence is considered.
Regex(r‘[.!?]+[>)\'}\]\"]*(\s|\n)+[\'\"{\[\(<]*[A-Za-z0-9]+’) is used for finding the sentence boundaries and words at the start of the sentence.

4. Mention your observation from the above set of operations in the report file.

Sentences make more sense in the order of unigram < bigram < trigram.

This is because the bigrams and trigrams consider the words surrounding in the corpus, connecting more similar bigrams and trigrams to form sentences. It captures the context and world knowledge. Bigrams considers 1st order Markov Assumption which states that the current term should only depend on the previous term while trigram considers 2nd order Markov Assumption which depends on 2 terms preceding the current term. Hence, trigrams capture more context as compared to bigram and bigram captures more compared to unigram which is independent of any previous term. But these won’t generate perfect sentences as N-grams cannot capture the long-distance dependencies(terms which are inter-related but separated by a large number of terms in between).

Similar assumptions have been taken for the rec.motorcyle corpus. Results are as follows:

<u>Results:</u>

Corpus: comp.graphics

Unigram:

image jpeg would file graphics images also use one available software data files format like

Bigrams:

1. the original is a few posts a lot of the same as a good choice is not
2. to the image processing and the following is the current version of a program that is available

Trigrams:

1. the egavgaadapter by addisonwesley you will have to be able to do the right thing with 24bit displays
2. the copierprinter has both a 486dx50 and a number of colors is set to 0xc018 acknowledgment i would

Corpus: rec.motocyles

Unigram:

writes article bike would one like get dod know go good well ride time right

Bigrams:

1. the first bike is a few of the bike and the same thing to the right side
2. to be a good place for the front of the road the other day i am not

Trigrams:

1. the areas you are not on the road and backroad a good bike to the right side
2. to treat others their property and their collective communications hub recmotorcycles will not be a squid if

<u>Part 2: Discriminative Model</u>

3.

Following steps have been followed:

- The user input sentence is preprocessed similar to the corpus.
- The words are tokenized to form bigrams.
- Then the log probabilities are calculated for each of the classes and one with the highest probability is returned as the class label.

4.

- Heuristic: The unigrams with the lowest frequencies(i.e. 1) are converted in <UNK> tags in the vocabulary.
- Then unigrams and bigrams are generated.
- Same steps are followed as in part 3.

Here, the probabilities with the <UNK> in it are overpowering the classification as the count of the words in the vocabulary with frequency 1(which are converted to <UNK>) is very high.

Example case:

```
Enter the sentence: hi girl waasssupp
('hi', '<UNK>') -7.67619544358
('<UNK>', '<UNK>') -2.79501587321
-10.4712113168
('girl', '<UNK>')
('hi', 'girl')
('girl', '<UNK>') -8.83695515733
('hi', 'girl') -8.83869681234
-17.6756519
```

To tackle this problem, one heuristic can be followed as the probabilities generated for the bigrams consisting <UNK> tag can be multiplied by 2(divided by 2 in case of normal probabilities).

References:

https://towardsdatascience.com/pre-processing-in-natural-language-machine-learning-898a84b8bd47
https://nlpforhackers.io/language-models/
https://www.kdnuggets.com/2018/03/text-data-preprocessing-walkthrough-python.html
https://www.usebackpack.com/resources/18980/download?1535371671