CS918

u1853897 Baihui ZHENG

Part A: Text Preprocessing

In order to read the whole json file, I used *readlines()* , because *read()* cannot load such a big file in one time. The following operation is setting regular expressions to filter the texts. By using *r'[^\s0-9A-Za-z]'* , *r'\b\w[1]\b'* , *r'\b\w[1]\b'* and *r'[A-Za-z]+://[^\s]*'* , the programme can filter all non-alphanumeric characters except spaces, remove words with only 1 character, remove numbers that are fully made of digits and remove URLs. In this programme, URLs are the first things be removed, then all non-alphanumeric characters except spaces are removed, next numbers that are fully made of digits with are cleaned , finally words with only 1 character are replaces by *""* .

For lemmatization, firstly this programme imported *WordNetLemmatizer()* function. Then using *pos_tag* to set POS_tag for every word. POS is mainly used to label the components of words in the text. In this programme, there are 5 kinds of POS, including NOUN, ADV, ADJ,VERB and unknow. *Lemmatize()* can lemmatize different kinds of words based on their types. This function is extremely useful, but it is really time consuming.

Part B: N-grams

Firstly, utilizing a list can store all words after text preprocessing. Then, it is convenient to calculate the number of tokens and the vocabulary size. *len()* and *set()* can separately gain the number of tokens and the vocabulary size of this corpus.

Nltk package provides a considerable number of methods for language processing. *nltk. word_tokenize()* can divide sentences into single tokens. After obtaining tokens, *nltk.collocations.TrigramCollocationFinder.from_words(tokens)* can discover trinary

phrases and sorting them, it typically builds a searcher using the function *from_words(). nbest()* can output the first 25 most common trigrams on the corpus.

With *positive-words.txt* and *negative-words.txt* , we can build a set to store positive and negative words. In order to compute the number of positive and negative words counts in the corpus, this programme compares each word with the set of positive and negative words. And counting the number of positive and negative stories.

Part C: Language models

At first, this programme trains a trigram language model. The process of training is storing words on the first 16000 rows of corpus, then preprocessing train sentences, next storing bigrams and trigrams on the first 16000 rows, finally using *Counter()* to convert bigrams and trigrams to Counter, which can speed up the running rate. After getting the language model, we want to create a sentence. This programme use a list to store sentence, so the first two elements should be *"is"* and *"this"* . Then we match bigram *(is,this)* with all trigrams, if matching successfully ,this programme will store the matching trigrams in a list. So the third element in the most common trigram of the list should be the next word following *"is"* and *"this".* We use a 8-times for loop to realise produce a sentence of 10 words.

To computing perplexity, we divide all the remaining rows into sentences. Then storing all trigrams in each sentence. Next, recording the occurrences of corresponding bigrams and trigrams on the first 16000 rows. Because some sentences are less than 3 words, we ignore them. Also some trigrams cannot be found in train_set, we use LAPLACE SMOOTHING. Finally applying the formula of perplexity can get the result.

The screenshots

```
In [9]: runfile('C:/Users/Administrator/Desktop/NLP/exercise1/final.py',
wdir='C:/Users/Administrator/Desktop/NLP/exercise1')
this corpus contains 10400 positive stories
this corpus contains 6941 negative stories
this corpus contains 176494 positive words
this corpus contains 143471 negative words
This corpus contains 5883766 tokens and 123107 vocabularies.
top 25 trigrams:  [('02jun16', '09jun16', '16jun16'), ('0749s', '290kmh',
'loris'), ('09jun16', '16jun16', '21jul16'), ('119x', '121x', '354x'),
('1250jkg1335', '1500ume1545', '1820dus'), ('16jun16', '21jul16', '28jul16'),
('1946s', 'gilda', '1935s'), ('1esrc', 'ufparentnodeinsertbeforee',
'fdocumentcreateelementscriptdocumentgetelementsbytagnamescript0'), ('1x8x',
'msaa', 'remappable'), ('21675s', 'aurelien', 'panis'), ('21jul16', '28jul16',
'04aug16'), ('22120s', 'jazeman', 'jaafar'), ('282de', 's68', 'f52jkam'),
('28lq6q', 'ii28lq6q', '5clqecqmde'), ('29860s', '0019s', '180844mph'),
('29910s', '0069s', '180542mph'), ('29anthony', 'warlow', 'neverlands'),
('2bromo2nitropropane13diol', '5bromo5nitro13dioxane',
'hydroxymethylglycinate3'), ('30018s', '0177s', '179892mph'), ('30044s', '0203s',
'179736mph'), ('30059s', '0218s', '179647mph'), ('30144s', '0004s', '179140mph'),
('30163s', '0023s', '179027mph'), ('30172s', '0331s', '178974mph'), ('30173s',
'0033s', '178968mph')]
['is', 'this', 'a', 'good', 'thing', 'the', 'answer', 'is', 'no', 'longer']
42090.347733125025
running time is 813.2545492649078 second
```