2.1. Preprocessing_w2v_100_tweets

October 18, 2017

```
In [2]: import pandas as pd
        import numpy as np

        import timeit
        import nltk

        import scipy.spatial.distance as distance

In [5]: df = pd.read_csv('datasetG.txt', sep="\n", names=['p1'])

In [6]: # Words are separated, creating a list of words instead of a string and then calculati
        df['p1'] = pd.Series([nltk.word_tokenize(x[0]) for x in df.itertuples(index=False)], in

In [7]: #I did it in a start to remove : , ... but then I remembered that they are punctuation
        #so it doesn't work remove he, to... print stopwords if you want to see them all.
        from nltk.corpus import stopwords
        stopwords = nltk.corpus.stopwords.words('english')
        df['p1'] = df['p1'].apply(lambda x: [item for item in x if item not in stopwords])

In [3]: #Word2Vec load in memory
        import gensim
        start_time = timeit.default_timer()

        model = gensim.models.KeyedVectors.load('../../quora/data/GoogleNews-vectors-negative3

        print(timeit.default_timer() - start_time)

33.81267497999988


In [5]: # For each document or tweet it generates a dict with all the words in the tweet and t
        def feature_vector(tweet, num_features, model):
            #print(tweet)
            words_not_founded = set()
            featureVec = np.zeros((num_features,), dtype="float32")
            vecTweet = {}

            for word in tweet:
                if word in model:
                    featureVec = np.add(featureVec, model[word])
                    vecTweet.update({word : featureVec})

                else:
                    words_not_founded.add(word)
            #print(words_not_founded)
            return vecTweet

        #feature_vector(df['p1'][0], 300, model)
```

```
In [6]: #Calculate the vector of each tweet and saving it in the df
        df['vector'] = [feature_vector(row[1], 300, model) for row in df.itertuples()]
```

```
In [8]: df.to_pickle('my_df.pickle')
```

```
In [ ]:
```

```
In [ ]:
```