CSE 654 HW3 REPORT

YUSUF PATOGLU - 151044027

SENTIMENT ANALYSIS OF TURKISH MOVIE REVIEWS

WITH WORD2VEC AND GAUSSIAN NAÏVE BAYES

EXPLANATION

The homework is about training a sentiment analysis with the help of Turkish sentences from Wikipedia and actual labeled movie reviews. We will use train a model with the help of unlabelled sentences(wikidump). This way we'll be using much more texts.

COMBINING FILES

Before training the data, preprocessing should be made. Also before giving our file to word2vec program we should be aware that our file doesn't contain weird symbols, newlines, tabs etc. I did this preprocessing in the file named file_operations.py. It fixes the common errors like file format, decoding/encoding errors etc.

The input files are named as follows: "test.csv" which is our labelled test set, "train.scv" which is our labelled training movie data set and finally the "trwiki" file which contains Turkish Wikipedia reviews. In my folder I only included 5% of actual trwiki dump in order to upload to Moodle. The scripts combines these files then outputs a space separated words with the file named with "word2vecinput".

The combination of three files (word2vecinput file) in this form at the end:

```
film çok vasat hiçbir özelligi yok tavsiye etmem eski trk filmlerini izlememi
cok guzeldi filmin icinde hissettim kendimi sonu biraz daha degisik olsa super
önemli bir kaniti hirschin genç yastaki ustaligi doga ve insanin lirik iliskisi
yasama arzusu rahatlikla izlenebilir gereksiz bir film we gereksiz oyuncular d
nna çok fazla dövüs sahnesine ragmen çok sikici bir film izlemesinizde olur yan
eyen arkadaslara tavsiyem sinemaya gidecek paraniz varsa paranizi bu filmle çöp
lerinden biriydi oldukça farkli olmus çok begendim ben filme dram aksiyon denmi
ederim gerçekten bu oyunculara layik olmayan bir film çekilmis bir iki aksiyon
fen daha gercekci yorumlar yazin bu film bu kadar puanida kesinlikle hak etmiy
```

FIGURE 1: CONTENT OF WORD2VEC INPUT FILE

GENERATING VECTORS WITH WORD2VEC

After creating the input file for word2vec program now we can create our vectors with it. I tried running word2vec program with different parameters but the default parameters with the vector size 100 gave the best results.

```
/Desktop/word2vec-master$ ./word2vec -train word2vecinput -output vectors -min-count = 1
Starting training using file word2vecinput
Vocab size: 64058
Words in train file: 562569
Alpha: 0.000106  Progress: 100.66%  Words/thread/sec: 221.91k  patoglu@ubuntu:~/papatoglu
patoglu@ubuntu:~/Desktop/word2vec-master$
```

FIGURE 2 : CREATING VECTORS WITH WORD2VEC

No we've created the "word" vectors but what we need is sentence vectors. There are several ways to calculate the feature vector of a sentence. These are:

1. Calculating the average of word vectors.
2. Calculating the average of word vectors with their TF-IDF values.

These calculations will make an impact on probability calculation. In homework it says provide atleast 2 ways but I only implemented the first one. It simply tokenizes the sentence and according to word's vector score calculating the average of a sentence. We will use these comment vectors while training our model.

```
kötüyse 0.002642 -0.004564 -0.001119 -0.003384 -0.003334 -0.000005 0.0029
çekmenin 0.004631 -0.000352 -0.007019 0.003903 -0.015771 -0.022751 -0.004
izlemenisi 0.010329 -0.026544 0.022713 -0.015573 0.028809 -0.025592 -0.00
kalmistim -0.017033 -0.006224 0.000951 -0.012175 -0.005261 0.005666 0.026
özelllikle 0.010105 -0.032191 0.045078 -0.007272 0.008146 -0.028305 0.008
önemliside -0.022158 -0.006802 0.000828 -0.011347 -0.037074 -0.039191 0.0
çagdan 0.005318 -0.021060 -0.018921 0.012120 0.006827 0.024330 -0.012424
reklama 0.009523 -0.026586 -0.006013 -0.025535 0.066915 -0.004067 0.01207
edindi -0.004580 -0.007601 -0.005284 0.028625 0.022903 0.021653 -0.014801
askla 0.004146 -0.004634 -0.001582 -0.000442 0.002086 0.001770 -0.001313
aglat -0.022772 0.014816 0.017567 0.002160 -0.029847 0.008301 0.005022 -0
```

FIGURE 3: THE OUTPUT FILE OF WORD2VEC PROGRAM

CONTINIOUS NAÏVE BAYES AND TESTING MODEL

Instead of using classical Naïve Bayes we will use Gaussian Naïve Bayes because our model is continuous. So we won't be doing counting as shown in the book.

(b) If A_k is continuous-valued, then we need to do a bit more work, but the calculation is pretty straightforward. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad (8.13)$$

FIGURE 4: GAUSSIAN NAIVE BAYES ON DAN JURAFSKY'S NLP BOOK

I couldn't implement this algorithm on my own so I used a library which does this calculation behind the scenes. A part of my code and the final result is like:

```
model = GaussianNB()
model.fit(np.array(sentence_vectors), np.array(labels))
prediction = model.predict(np.array(sentence_vectors_test))
print(f1_score(np.array(test_labels),prediction, average = 'micro'))
```

FIGURE 5: TESTING THE MODEL

Finally my accuracy was always between 0.65-0.67:

```
PS C:\Users\patog\One
0.6669167291822956
```

FIGURE 6: FINAL SCORE OF THE MODEL