CS 584: Data Mining (HW1)

Md. Ridwan Hossain Talukder
miner2 ID: Luffy
Rank on miner2: 16
Accuracy: 0.86

I. Problem Statement

In this homework we are trying to infer sentiment (or polarity) from free-form review text submitted for a range of products in e-commerce application. For this purpose I have implemented a k-Nearest Neighbor classifier to predict the sentiment for the reviews for various products.

II. Methodology

A. Text Preprocessing

Text data are unstructured, especially when it comes to sentiment analysis from large set of reviews. So before extracting features from the text data we need to clean and pre-process the texts so that we remove any noises present in the data. To clean the data following steps were used:

1) Removing punctuation and whitespaces: All the text were made lower case and removed punctuation and white spaces. The process is done by tokenizing the text using the NLTK's $word\_tokenize$ library function and keeping only the alphanumeric words in the text.

2) Removing stop words: Filtering through a list of stop words sometimes improves performance while processing text data, on the other hand there might be some lose of information due to ineffective word removal. Like, as we are analyzing sentiment from raw text, removing frequent words like a, an, the (articles) or (on, by, here) type words may be useful but at the same time if we remove some frequent words like no, not, don't from the the text data, it may lose some important information. So I did not use the default stop words from the nltk library rather constructed a list of frequent stop words where I tried to keep the words, removing which may be helpful.

3) Lemmatization or Stemming: I have tried some of the Stemming techniques (LancasterStemmer, PorterStemmer, and SnowballStemmer from the NLTK library) and Lemmatization technique (WordNetLemmatizer from NLTK) to produce the morphological variants of a root word. While performing stemming was a bit faster but that did improve a lot in terms of accuracy over lemmatized the word. On the otherhand for some cases it was better using no stemming or no lemmatization of the word. So I after I figured out the best k-value for my classifier and the values for the feature were determined, I figured out using PorterStemmer was performing a bit better than using other stemming process and lemmatization. So I used the PorterStemmer library for stemming the words.

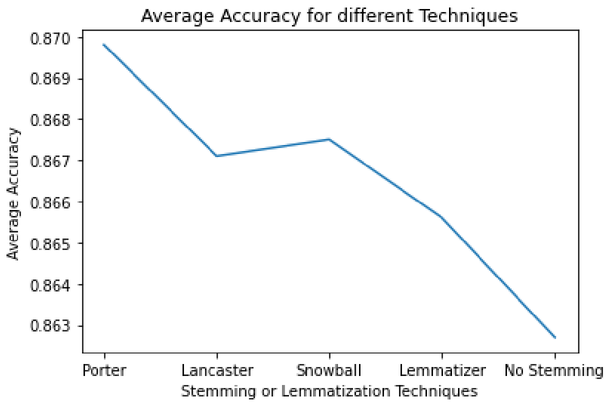

Fig. 1: Average Accuracy for different Techniques (K=171)

4) Parts of Speech Tagging: I also used the POS (Parts of speech) tagging (from NLTK library), to see whether that improves the overall accuracy if we only choose the Adjectives (as sentiments are mainly coming from adjectives). But it reduced the overall average accuracy to I did not use it in my final prediction.

B. Feature Extraction and selection of the parameters

I have used the BOW(Bag of words) model for feature extraction, where we have a vocabulary created from the list of text documents that we have as a training data. Instead of using only word count with CountVectorizer I have used the TF-IDF(Term Frequency-Inverse Document Frequency) technique for information retrieval from the bag of words. As TF-IDF value increases in proportion to the number of times a word appears in the document but is often offset by the frequency of the word in the corpus, which helps to adjust with respect to the fact that some words appear more frequently in general. So, I have used the TfidfVectorizer from the sci-kit learn library to create the matrix of TF-IDF features from the processed text documents. I have used 4 parameters (max_features, max_df, min_df, ngram_range) to construct the vector representation of the text data. I choose the values for the parameters in two steps. First, before finalizing the values of the parameters I determined the best k value based on trying the default combination of the parameters with different k values using k-fold cross validation and once the k value was determined then I tried different combinations of the parameters to select the best combination that gives best average accuracy by running the k-fold cross validation again on the training data set. To choose the value of max_features we take a random value from 1000 to 100000 and saw that taking the max_features 50000 we got the best accuracy. On the other hand we tried some values from range (0.7 to 1.0) for max_df and (0 to 5) for min_df. ngram_range was taken from (1,1), (1,2), (2,2), (2,3) and (3,3). The values that gave the best average accuracy with the selected k value is max_df: 0.9, min_df: 1, ngram_range: (2,3).

TABLE I: Average Accuracy for some combinations of selected parameters (kept only with significant improvement) with K-value 101

max_features	max_df	min_df	n_gram_range	avg. acc.
None	1.0	0	(1,1)	0.815
2000	0.9	1	(1,2)	0.782
5000	0.9	0	(2,2)	0.825
20000	0.9	0	(2,2)	0.847
50000	0.9	1	(2,2)	0.853
50000	0.9	0	(2,3)	0.857

C. Implementation of K-Nearest Neighbour classifier

We have implemented the simplest version of KNN classifier. To get the nearest neighbours we have taken the distance metrics and similarity metrics in equation. To calculate the distance from two vectors we used the manhattan distance and euclidian distance and also used the cosine similarity to find the similarities between two vectors. We choose the metrics from these three that give us the highest average accuracy for the set up (choosing optimal k, best features) of our classifier. In our experiment we get the best accuracy using the cosine similarities so we used the cosine similarities metrics to predict sentiment on the test data set.

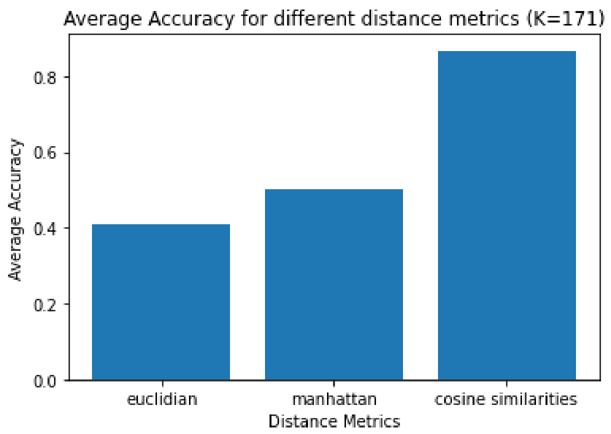

Fig. 2: Selecting k-value using the avg accuracy obtained from k-fold cross validation (using current best set up)

D. Finding the optimal k value

1) K-fold validation (k=5): To get the optimal value of k we used the k-fold(k=5) cross validation model. It gives us the value of k that we can use to select the number of neighbours we should choose to predict the value of the new data point from the test data set. First we choose the best k value for a default feature parameters which was 101 (we choose the value by plotting the data and observing the pattern). Once preliminary k-value was selected then again we used the k-fold validation to choose the best parameters and then again used it to choose our final k-value. Initially for k=101 we got the highest average accuracy, however after parameter tuning we got our latest k value in 171.

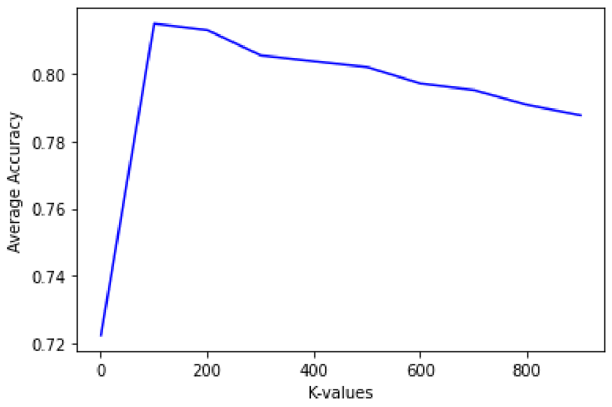

Fig. 3: Selecting k-value using the avg accuracy obtained from k-fold cross validation using default parameters (max_features = None, max_df = 1.0, min_df = 0, ngram_range = (1,1)

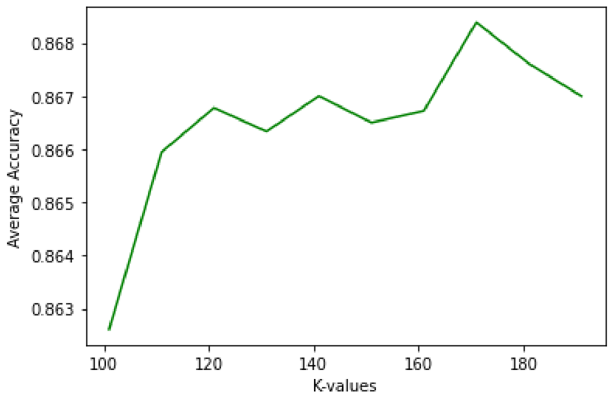

Fig. 4: Selecting k-value using the avg accuracy obtained from k-fold cross validation using determined parameters (max_features = 50000, max_df = 0.9, min_df = 1, ngram_range = (2,3)

2) Voting Techniques: We used both majority voting and weighted voting to check which performs best. For majority voting it is as simple as counting the total number of votes while weighted voting also counts the distance between the data points as well. But as both the measure performs closely (majority voting performs slightly better) we selected the majority voting to predict the sentiment of the data point.

III. Results

Our model gave an average accuracy of 86.9% on training data when we cross validated the accuracy with k-fold(k=5)

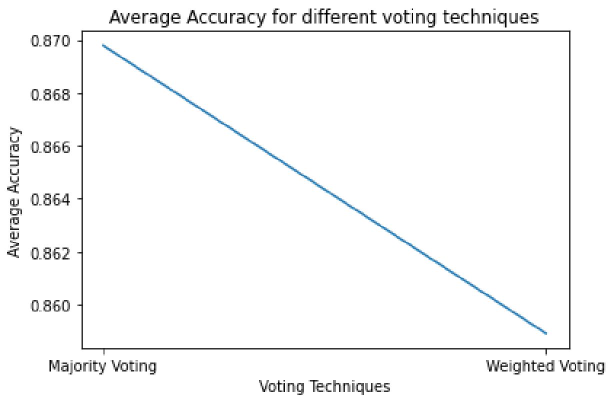

Fig. 5: Average Accuracy for different voting techniques (K=171)

cross validation. However, it performed closely(0.86 accuracy on miner2) on the test data as well (approximately as the percentage on the miner is approximate at the time of writing this report). In a nutshell we used the TF-IDF technique on pre processed text data with the parameters ((max_features = 50000, max_df = 0.9, min_df = 1, ngram_range = (2,3)) and our implementation of KNN uses the cosine similarity metric to choose the nearest neighbour(s) and to predict the value from the k nearest neighbour we used majority voting. There are some interesting observation in terms of experimenting with the k-value with default parameters and selected parameters. First of all, when we choose the TF-IDF technique with no stop word removal, or text pre processing and using default parameter that the maximum feature is None and maximum data frequency is 1.0, minimum data frequency is 0 and ngram range is (1,1) the accuracy was below 75%, and but tuning the max feature parameter and ngram collective increased the accuracy to a great extent. Other parameters also contributed but these two are the main factors to increase around 5-7%. And I belive that's because instead of unigram we have used bigrams and trigrams which reduced the bias towards a positive adjective with a negative determiner (eg: not good, don't recommend) and increased the accuracy.