Data Mining – Final Project

Beni Broohm

I. Description of the problem

We have some product reviews coming from Amazon database and we would like to predict the user rating according to the review. That is the main idea. But precisely, we want to build a classification model that can give good prediction given a review (which we consider as a list of words). Now, how do we build that?

II. Modelling the solution

We considered two ways of solving this problem :

- Bag of words approach considered as the basic solution
- Our model which uses a matrix of semantic meaning instead of only word frequency in the text.

We used two main algorithms for classification :

- Knn with the Minkowski metric : the idea here is that we get the k closest neighbours of a test sample and we choose the most recurrent class as the class of our sample.
- Random Forest : the idea is that we have many decision trees, each of them trained with a random subsample. We get a random subsample on which we build a classification tree. The sample that needs classification takes the most recurrent class from the trees.

1. Bag of words approach

The main idea behind Bag of words/features is to consider the words' frequencies in the text and use that as a characteristic on which we can lean to make classification.
First, we get vectors of features by building a dictionary of the filtered and stemmed text and for each row (clean reviews and user rating), we create a vector of the size (d) of the dictionary. For each word in the review, we increment the value at the corresponding index. We then use a classification algorithm to fit the vectors to the user rating. This gives us a model which can predict the user rating according to a vector of features.

2. Our model approach

Instead of making classification based only on word frequency, we introduce the semantic of the words. So basically, we have a dictionary which maps a word to a vector of size 300 (200 in the file word2vec.txt) that represents the meaning of the word. We then make an average to get the general meaning of the review and according to that, we do classification.
Precisely, we have a dictionary (word2vec) that contains words and the numerical representation of their meaning. So, synonyms have a closer numerical representation. For every review, we compute the mean of the meanings (mean of the meaning vector). We use our algorithms to make a model that can predict user rating according to a vector of meaning.

III. Results

According to our intuition, our model should easily beat the base model since it does not consider the semantic of the words but we were wrong. The Bag of words model beat our model 60% of the time. Let's see the following results. The table 1 shows us the performance of our algorithms with the models.

	Base model			My model		
File	Random Forest	Knn, k = 1	Knn, k = 8	Random Forest	Knn, k = 1	Knn, k = 8
always	63.49	50	62.7	64.28	50	61.9
gillette	59.56	40.43	53.15	55.81	40.31	50
oral-b	67.14	58.32	65.72	65.94	50.92	62.57
pantene	49.61	42.52	44.09	48.82	40.16	41.73
tampax	74	70	72	74	44	68

Table 1 : Performance comparison of the models (%)

We can see that the base model gives better results overall. Our model is slightly less effective but we have almost the same results for the different algorithms. We can also see that the Random Forest Classifier gives the best performance.
In the table 2 below, we have the McNemar test that gives us the number of times one model beats the other. And the result shows us that :

- The base model is the clear winner with 60% of the executions
- Our model wins 30% of the time
- Both models get the same results 10% of the time

	Base model wins	Base model loses	Row total
My model wins	2	6	8
My model loses	12	/	12
Column total	14	6	20

Table 2 : McNemar's test results

From our point of view, the main reasons why the base model is more efficient are as followed :

- The average vector does not give us the full information for classification. Building an average vector means compressing the available data for a better usage.
- When using the semantic meaning with average, we lose the order of the words. This means our model does not differ a lot from the base model since we do not consider the order of the words. It explains why the performances of the models are similar.
- The word2vec file does not contain data for every word in the datasets.

The reason why the Random Forest Classifier is better is that we use more predictors. For classification, we used 200 Classification Tree, which is a lot better than using 8 or 12 peers to make the classification.

IV. Problems/Solutions

We faced a few non-critical problems :

- Data not clean : there were some unwanted items in the datasets such as HTML tags and some stuff. We fixed that by deleting those items.
- The problem itself : understanding the problem and thinking about a solution was not easy. But it was more accessible when we searched and understood similar projects on Internet.