Natural Language Processing Section

Lucas Rodriguez

1 NLP

1.1 Dataset

To test our method in a natural language processing setting, we used the *20 Newsgroups* dataset. This set contains about 20,000 documents sourced from small news articles and grouped into 20 different categories. These categories include different sports, politics, and other news related topics.

These documents were used to create an input of 100 bags, with 50 positive bags and 50 negatives bags. Each bag included 50 instances, where each instance was a document from the dataset. Additional positive instances were chosen at 0.03 positivity rate in the positive bags.

The documents or instances are described by a feature vector that can be created in different ways. The main idea is to transform the words in a document to a format that a machine learning algorithm can understand. In this case, the feature vector is a list of 200 numbers or features created in three different ways. The feature extraction methods we tested were Bag of Words, TF-IDF, and Doc2Vec.

1.2 Feature Extraction

The first step we took in creating feature vectors for each document was removing the metadata and stop words. This resulted in a vector of the words in the document that could be used in the next step.

For Bag of Words (BoW) and TF-IDF, we put together all the words in our documents and noted their frequency. From there, we created a vocabulary of the 200 most frequent words. For BoW, the features for each document then became the number of times that each word in the vocabulary appeared in the document. TF-IDF feature vectors were created in a very similar way, but some extra calculations were done on the words and their frequency to get a better idea of the importance of a word.

Doc2Vec is a feature extraction method based on the Word2Vec algorithm. We implemented it in our parsing function using the *gensim* library.

After the feature vectors were created using one of these methods, we standardized the data. This gave us slightly better results, especially with the Doc2Vec extraction method.

1.3 Results

	BoW		TF-IDF		Doc2Vec	
	ACC	BACC	ACC	BACC	ACC	BACC
MIsvm	0.56 ± 0.169	0.568 ± 0.095	0.51 ± 0.187	0.508 ± 0.02	0.5 ± 0.193	0.5 ± 0.00
sbmil	0.52 ± 0.18	0.516 ± 0.101	0.501 ± 0.157	0.5 ± 0.00	0.501 ± 0.157	0.5 ± 0.00
smil	0.491 ± 0.138	0.508 ± 0.091	0.542 ± 0.107	0.541 ± 0.089	0.523 ± 0.155	0.534 ± 0.137
nsk	0.52 ± 0.098	0.521 ± 0.092	0.632 ± 0.097	0.643 ± 0.098	0.489 ± 0.118	0.507 ± 0.12
stk	0.551 ± 0.133	0.545 ± 0.137	0.511 ± 0.07	0.528 ± 0.084	0.398 ± 0.117	0.517 ± 0.027
ours	0.459 ± 0.082	0.486 ± 0.086	0.541 ± 0.147	0.547 ± 0.149	0.54 ± 0.064	0.55 ± 0.067

The results shown above are the outcome of running multiple machine learning methods on the 20 Newsgroups dataset. In this specific instance, we used the 'rec.motorcycles' group as the focus group. However, as we saw in previous tests, the focus group does not alter the results substantially. In fact, the results vary quite a bit from test to test using the same group because the parser function creates a new input dataset each time. The most important part is that all the machine learning methods are tested against each other using the same input dataset, which is how we implemented the testing process.

Our method has comparable performance to the other methods, especially with TF-IDF and Doc2Vec. In some cases, other methods did not perform correctly and marked all bags as positive bags. We had to remove some methods from our testing because they always showed this behavior. However, this never happened with our method.

In terms of speed, our method took about 1.5 minutes on average to complete. MIsvm took about 30 seconds, sbmil took about 40 seconds, smil took about 5 seconds, nsk took about 1 second, and stk took about 1 second. Although our method was slower than the other methods we used to test, it was not the slowest among all the methods we originally used. For example, misvm was taking upwards of 4 minutes to complete, on average.

We also tuned the parameters of our method to find the optimal set up for this dataset. For TF-IDF, we found that $(c = 0.01, \mu = 1e-4, \rho = 1.02)$ gave us the best, most consistent results. We did not try tuning the parameters as much for BoW or Doc2Vec, but we found that the default parameters, $(c = 0.1, \mu = 1e-3, \rho = 1.2)$, worked relatively well.

In terms of which feature extraction performed the best, we saw that TF-IDF had the best results in comparison to other methods. Doc2Vec and BoW also performed similarly, but did not consistently give good results like TF-IDF did. This may have to do with how we tuned the parameters, although it probably didn't have too big of an impact on performance.

1.4 Shortcomings and Future Research

There were some shortcomings in our testing that we could have tried to improve with more time. The way we created the vocabulary using the frequency of the words may have made our input less reliable. Instead, we should have used a different method, like information gain ranking. We also did not try altering the number of bags, instances, or features in our input which could have given us better results. Since our vocabulary and feature extraction were a shortcoming, maybe using more features (> 200) would have improved our results. One more thing that we could try to alter is our model's tuning parameters. We tested the tuning parameters well for TF-IDF, but with more time, we would have found the best parameters for BoW and Doc2Vec.

However, our results were not disappointing. Our method produced comparable results to all the methods we tested. The run time of our method was also respectable in comparison to some other methods we tried.