Hacettepe University

Computer Science and Engineering Department

Name and Surname : Mehmet Alperen Özçelik

Identity Number : 21992995

Course : AIN442

Experiment : Assignment 5

Date Due : 08.01.202

Advisor : Hayriye Çelikbilek

e-mail : b21992995@cs.hacettepe.edu.tr

1 - Introduction

2 - Data

3 - Method

4 - Development

4.1 Plan

4.2 Analysis

4.3 Design

4.4 Implementation

4.5 Programmer Catalog

4.6 User Catalog

5 - Results, Discussion and Conclusion

References

1-INTRODUCTION

This assignment is a project in which a Turkish dataset is applied in order to understand and reinforce the use of Word2Vec and LSA. Our main goal in our project is to make our dataset meaningful and apply Word2Vec and LSA to the given texts, to guess which of the 7 categories the texts belong to and make the model ready for test sets for the future.

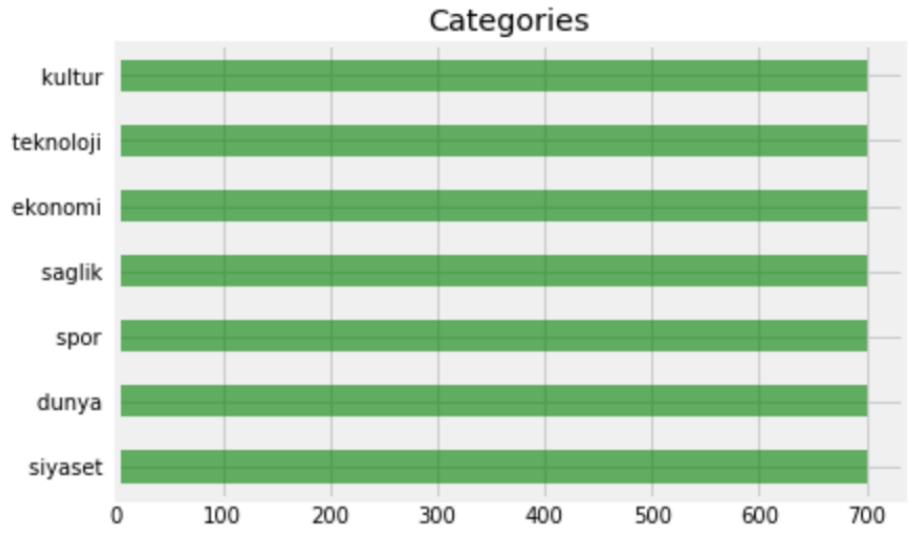
2-DATA

Our material is a file named turkish_dataset.csv containing the dataset we used throughout the project. There are 2 attributes: text and category.

And there are 7 categories in total:

- kultur
- teknoloji
- ekonomi
- saglik
- spor
- dunya
- siyaset

Categories of Texts:



As it can be seen, 700 examples of each of 7 categories are included in the dataset, and this model is important to avoid overfitting.

The image after converting the csv file into dataframe:

```
Length: 4900
```

	category	text
0	siyaset	3 milyon ile ön seçim vaadi mhp nin 10 olağan...
1	siyaset	mesut_yılmaz yüce_divan da ceza alabilirdi pr...
2	siyaset	disko lar kaldırılıyor başbakan_yardımcısı ar...
3	siyaset	sarıgül anayasa_mahkemesi ne gidiyor mustafa_...
4	siyaset	erdoğan idamın bir haklılık sebebi var demek ...

3–METHOD

To summarize, I first tokenized the text data we have and applied the classical NLP preprocess steps, which I will explain in more detail below. First, I used TF-IDF Vectorizer as a vectorizer and after printing the word clouds of the categories, I combined them with the LSA model and tested them on Naive Bayes, Logistic Regression, Decision Tree and Random Forest Classifiers.

Then I combined it with the LSA model using CountVectorizer(Bag of Words) as a vectorizer and tried it in 4 different classifiers.

Afterwards, I first put the array I obtained by using the Word2Vec process alone into the models, and finally, I combined Word2Vec and LSA and tried in 4 different models again.

As a result, I got 16 different models and by choosing the highest model, I applied hyperparameter tuning on it and brought my best model to the best version and completed the project by printing the conclusion matrix as a Heatmap.

4-DEVELOPMENT

4.1 Plan:

The general plan of my project was to complete the project by applying basic nlp preprocesses to my data, after tokenizing it, applying several vectorizers, LSA and finding the appropriate classifier. Since the language of the dataset in the project is Turkish, it is difficult to normalize and discard stopwords.

For this, I first used the "zemberek" library. Zemberek library has very nice functions for Turkish language processing such as sentence normalizer, tokenizer and spell checker.

First of all, I tried to use the texts in the dataset directly in the sentence normalizer function, but my computer requirements were not sufficient even if I did not run them in Google Colab or jupyter notebook.

Then I came up with a different solution in my mind and I tokenized the sentences with the tokenizer function of the zemberek library, then again using the spell checker function from the zemberek library, I selected the first word suggested by the library for each word and changed it in the text.

The library did not have a stop words dictionary, so in addition to zemberek, I removed the Turkish stop words from the corpus with the help of the nltk library.

Apart from the functions of these two libraries, I also changed the words to lowercase with the help of a small function and removed the punctuation marks in the texts.

4.2 Analysis:

When we looked at the features of data, there were only 1 column that I could use as attribute, the text column. As output, we had to get the category part. Since the texts are given in Turkish, I had to put them in a Turkish preprocessing process. Then I had to tokenize and try different vectorizes and different models to prepare the most suitable model for the problem.

4.3 Design:

There are several methods we can use for preprocessing after we tokenize the sentences. As far as I know, these methods are:

- Tokenize
- Convert all characters to lowercase
- Remove punctuation
- Remove Stopwords
- Use suggested word with Spell Checker Function
- Convert tokens to sentence

Since it is difficult to find a library for Turkish NLP Preprocessing, our teacher first suggested the “zemberek” library for us to use. Although zemberek is very successful in normalizing the sentences, I could not use the zemberek library because I had systemic problems while normalizing the texts in our dataset. Instead, as I explained above, I used the spell checker function from the zemberek library to replace the wrong words with suggested words and finally put them back into sentences.

Apart from these methods, I have classifier methods that I will explain in detail below, and I have methods that I use in the word cloud parts.

4.4 Implementation:

4.4.1 Latent Sentiment Analysis:

After applying preprocess to my texts, I first decided to use LSA. When using LSA, I could use count vectorizer or TF-IDF vectorizer as a vectorizer. I wanted to try both.

First, I used TF-IDF vectorizer and suppressed the wordcloud of the most important words while determining their categories. You can see the wordcloud of the technology category as an example below.

Worcloud of “teknoloji” category

Then I tried the 4 classifiers I've used so far to find the most suitable classifier. These classifiers were: Naive Bayes, Logistic Regression, Decision Tree and Random Forest Classifier. The accuracy scores are below:

```
Naive Bayes -> Accuracy: %69.08
Logistic Regression -> Accuracy: %79.18
Decision Tree -> Accuracy: %75.20
RandomForestClassifier -> Accuracy: %82.45
```

As you can see, Random Forest Classifier gave higher accuracy score in LSA with TF-IDF Vectorizer.

Afterwards, I decided to try LSA with Count Vectorizer this time, and after creating the Bag of Words of the model, I calculated the accuracy scores in 4 classifiers again and the servers are below:

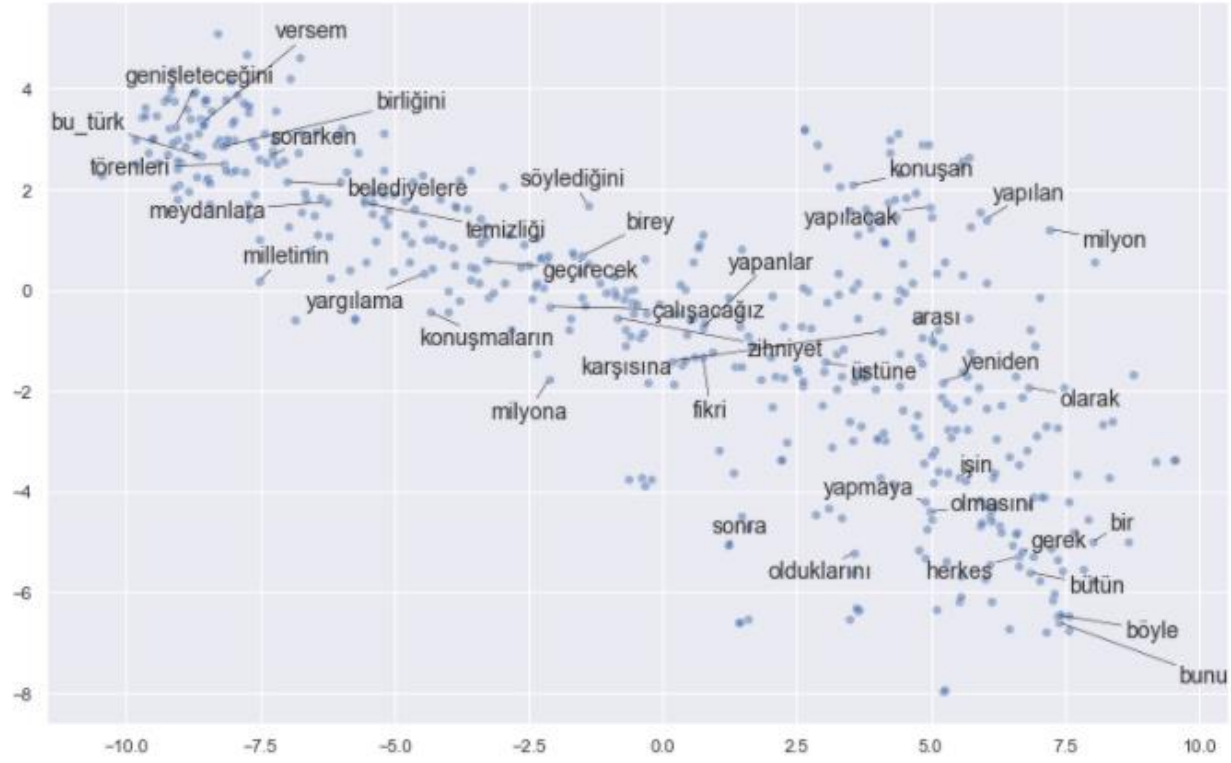
```
Naive Bayes -> Accuracy: %33.47
Logistic Regression -> Accuracy: %49.80
Decision Tree -> Accuracy: %51.12
RandomForestClassifier -> Accuracy: %61.02
```

I noted that my most suitable classifier for now is the Random Forest Classifier and I got better results with TF-IDF Vectorizer and moved on to the Word2Vec stage.

4.4.2 Word2Vec:

After my experiments with LSA were over, it was time for the Word2Vec model. First of all, I created a vocabulary using the words in my tokenized texts. Afterwards, I tried to understand the working principle of Word2Vec by looking at the most similar words of some words and the similarity ratios of the two words.

Then I visualize the representation of the word vectors in space with the TSNE library.



I created a dataframe in the shape (4900,100) by looking at the word 100, which is the closest similarity of the words in all the sentences, and now my Word2Vec model is ready to be tested with various classifiers.

After measuring the accuracy scores in 4 different classifiers, I realized that I still did not get as high accuracy as the old model, the accuracy scores are given in the image below.

```
Naive Bayes -> Accuracy: %48.67
Logistic Regression -> Accuracy: %63.67
Decision Tree -> Accuracy: %57.24
RandomForestClassifier -> Accuracy: %67.55
```

4.4.3 Word2Vec and LSA Model:

The last model I could do was to use the Word2Vec and LSA model together. For this, I created a new set by concatanating the dataframe in shape (4900,100) that I obtained from Word2Vec with the array (4900,7) obtained from the LSA model, but the model naturally gave the result by using Word2Vec alone due to its 100/107 ratio.

```
Naive Bayes -> Accuracy: %48.67
Logistic Regression -> Accuracy: %63.67
Decision Tree -> Accuracy: %53.57
RandomForestClassifier -> Accuracy: %66.63
```

As a result of all these trials, my best model was the combination of Random Forest Classifier, TF-IDF Vectorizer and LSA. The best score I got is 82.96.

All that's left is to do hyperparameter tuning for this model.

4.5 Programmer Catalog:

The tutorial shared with us in our AIN442 course was very instructive and I can say that I spent about 7 hours because I implemented my codes with a very similar design.

I mentioned above that I had difficulties in implementing the Word2Vec part, and finding how to use Word2Vec in model part I spent about 5 hours.

The truth is, I didn't have a test set to test my model. I separated 20 percent of it as test data with the classical train-test split method, but it would be more accurate to call this data validation data. it took some time to find the best classifier here and to find the best parameters to give while trying the classifiers, it took about 4-5 hours.

Finally, it took about 5-6 hours to report them with pictures.

My suggestion to other programmers is that Random Forest Classifier gave much better results than other classifiers in my dataset. I think it should be tried in NLP projects. The preprocessing parts is also of great importance for such language processing projects.

4.6 User Catalog:

Any user doing NLP project can use most of my code. Since the source code part I gave above is very simple and clear, I don't think there will be anything confusing about its use.

- ✓ User can use my tokenize and preprocess methods applied to any dataset which is Turkish or just change a little bit to preprocess other languages datasets.
- ✓ Count vectorizer and TF-IDF part is already a direct library function, so it can be imported directly.
- ✓ The user can use my LSA and Word2Vec code parts to try it on their own projects its too simple to understand.
- ✓ The user can test the 4 classifiers I use to find the highest accuracy and select the most suitable classifier.
- ✓ Finally, they can directly use the word cloud processing part by using the tokenized version of their own words and print their own word clouds.

5 - RESULTS, DİSCUSSION AND CONCLUSION

First of all, I realized how important the preprocessing process I performed with the mainspring library and the nltk library was. Because without preprocessing, I could achieve a maximum accuracy score of around 76 percent, while thanks to preprocessing, I achieved an accuracy score of around 82 percent.

I tried 16 different models in total and the most successful model was the LSA model working with TF-IDF Vectorizer. The best accuracy I get is **82.45**. LSA model provided a significant increase in acccuracy across this dataset, but I did not get the efficiency I expected from Word2Vec. Maybe when combining Word2Vec and LSA, the model can be concatanated in different ways, resulting in higher accuracy.

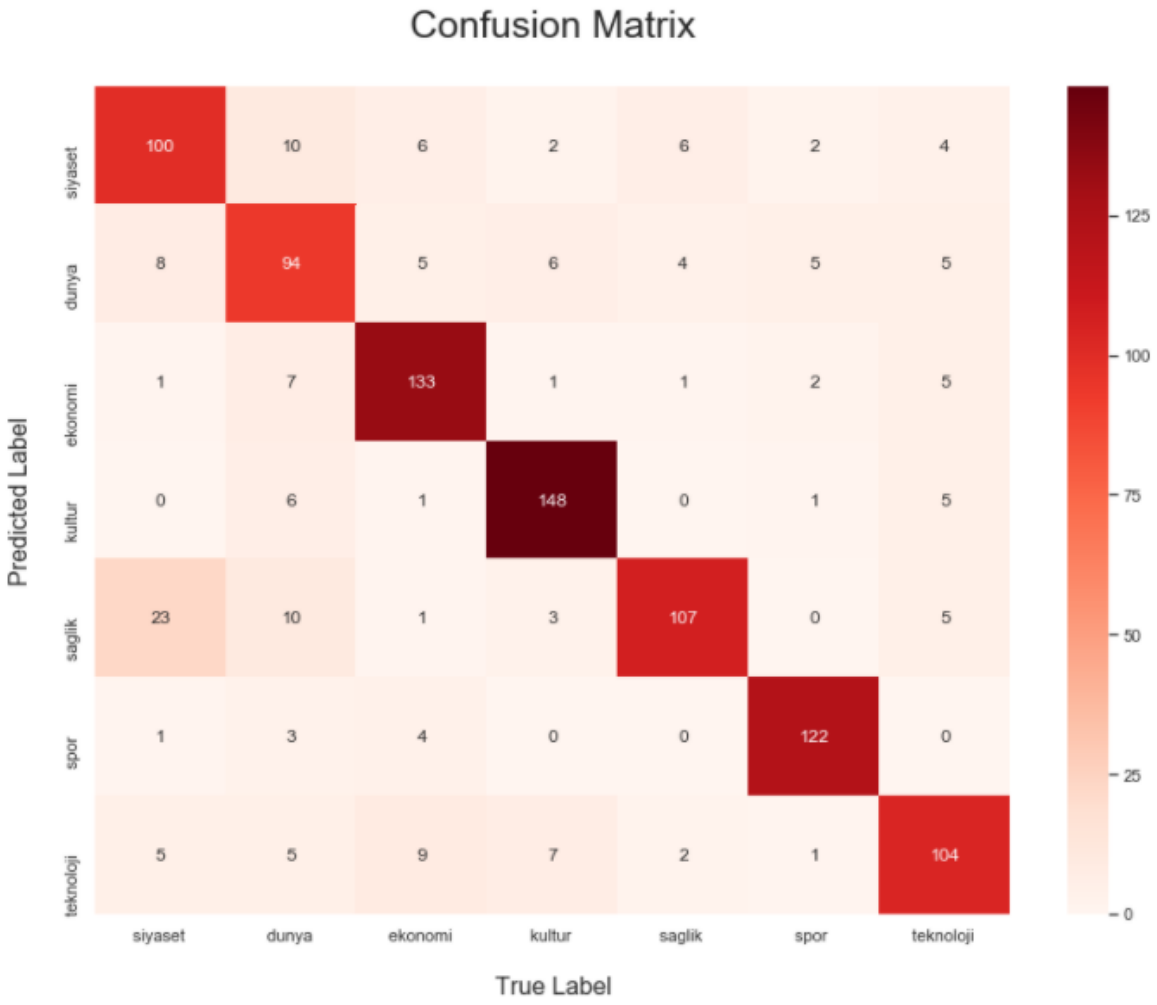
I obtained a model by combining them in the simplest way, apart from that, Random Forest Classifier was the best classifier by far, and Random Forest Classifier was again the most suitable model in my project in the previous assignment. I think Random Forest Classifier is a successful classifier in NLP projects.

Finally I applied hyperparameter tuning to Random Forest Classifier with RandomizedSearchCV to get better results. I use the confusion matrix created by my model as a heatmap with the best p arameters. I showed it, you can find its image below and my best parameters in Random Forest C lassifier was:

Best parameters:

```
{'n_estimators': 1000,
 'min_samples_split': 5,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 110,
 'bootstrap': True}
```

Confusion Matrix:



REFERENCES:

- https://towardsdatascience.com/using-word2vec-to-analyze-news-headlines-and-predict-article-success-cdeda5f14751
- https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
- https://www.kaggle.com/pierremegret/gensim-word2vec-tutorial
- https://github.com/Loodos/zemberek-python
- https://towardsdatascience.com/google-news-and-leo-tolstoy-visualizing-word2vec-word-embeddings-with-t-sne-11558d8bd4d
- https://github.com/xiamx/node-nltk-stopwords
- AIN 442 Lecture notes