CENG464 TEXT MINING

Term Project Report

Hazim Alper ATA – 260201044
Ertuğrul DEMİR - 260201059

06/01/2023

Content

A.1 Specific Preprocessing Methods

- Explain why you prefer these specific preprocessing methods in detail, which feature of the data made you think to apply these processes?

In our project we decided to use 5 different preprocesses 6 times. These preprocessess are in order:

1. Punctuation
2. Word Tokenization
3. Removing Duplicate Letters
4. Removing Stopwords
5. Verb Lemmatization
6. Noun Lemmatization

Because in data there are plural words, words that have duplicate letters, url link, nonsense punctuation, unnecessary words etc. So, we applied these steps.

Punctuation:

In punctuation we used default python string library punctuations. We iterate the letters and if it's in this regex we replace it with space.

```
r"""!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"""
```

Default Python string punctuation regex

Word Tokenization:

In word tokenization we used NLTKs *word_tokenize* function. We give the text in each file. We store them in a nested array.

Removing Duplicate Letters:

We realize the data has some words like 'heeellloooo'. We need to convert it to 'hello'. To do this process we use regex. If the same letter is repeated at least 2 times in a row, we reduce their repetitions to 2. If we continue the same example, 'heeellloooo' will turn to 'heelloo'. Now it's closer to 'hello'.

```
rx = re.compile(r'([^\W\d_])\1{2,}')
word = re.sub(r'[^\W\d_]+',
              lambda x: Word(rx.sub(r'\1', x.group())).correct() if rx.search(x.group()) else x.group(),
              word)
```

Why do we have this process ? Because we want to upgrade lemmatization performance.

Removing Stopwords:

In removing stopwords we use NLTK corpus English stopwords. We create the new word list that hasn't stopwords.

i	me	my	myself	we	our	ours	ourselves	you	you're	you've	you'll	you'd	your	yours	yourself	yourselves	he	him	his
himself	she	she's	her	hers	herself	it	it's	its	itself	they	them	their	theirs	themselves	what	which	who	whom	this
that	that'll	these	those	am	is	are	was	were	be	been	being	have	has	had	having	do	does	did	doing
a	an	the	and	but	if	or	because	as	until	while	of	at	by	for	with	about	against	between	into
through	during	before	after	above	below	to	from	up	down	in	out	on	off	over	under	again	further	then	once
here	there	when	where	why	how	all	any	both	each	few	more	most	other	some	such	no	nor	not	only
own	same	so	than	too	very	s	t	can	will	just	don	don't	should	should've	now	d	ll	m	o
re	ve	y	ain	aren	aren't	couldn	couldn't	didn	didn't	doesn	doesn't	hadn	hadn't	hasn	hasn't	haven	haven't	isn	isn't
ma	mightn	mightn't	mustn	mustn't	needn	needn't	shan	shan't	shouldn	shouldn't	wasn	wasn't	weren	weren't	won	won't	wouldn	wouldn't	

NLTK English stopwords

Lemmatization:

In punctuation we used *WordNetLemmatizer*. Normally, its lemmatizer function needs two values. First is a word that will be lemmatized, second is Part Of Speech(pos) that has default value is **n** (noun).

```python
def lemmatize(self, word: str, pos: str = "n") -> str:
    """Lemmatize `word` using WordNet's built-in morphy function.
    Returns the input word unchanged if it cannot be found in WordNet.

    :param word: The input word to lemmatize.
    :type word: str
    :param pos: The Part Of Speech tag. Valid options are `"n"` for nouns,
        `"v"` for verbs, `"a"` for adjectives, `"r"` for adverbs and `"s"`
        for satellite adjectives.
    :param pos: str
    :return: The lemma of `word`, for the given `pos`.
    """
    lemmas = wn._morphy(word, pos)
    return min(lemmas, key=len) if lemmas else word
```

WordNetLemmatizer lemmatize function

We apply 2 Lemmatization processes. One of them for nouns, the other one for verbs. This situation is explained very well with examples.

```
plot,two,teen,couple,go,church,party,drink,drive,get,accident,one,guy,dy,girlfriend,continues,see,life,nightr
happy,bastard,quick,movie,review,damn,y2k,bug,got,head,start,movie,starring,jamie,lee,curtis,another,baldwin,
movie,like,make,jaded,movie,viewer,thankful,invention,timex,indiglo,watch,based,late,1960,television,show,nar
quest,camelot,warner,bros,first,feature,length,fully,animated,attempt,steal,clout,disney,cartoon,empire,mouse
synopsis,mentally,unstable,man,undergoing,psychotherapy,save,boy,potentially,fatal,accident,fall,love,boy,mot
capsule,2176,planet,mar,police,taking,custody,accused,murderer,face,title,menace,lot,fighting,whole,lot,story
ask,8mm,eight,millimeter,really,wholesome,surveillance,man,loses,sight,value,becoming,enmeshed,seedy,sleazy,
exactly,long,movie,felt,even,nine,laugh,nine,month,terrible,mess,movie,starring,terrible,mess,man,mr,hugh,gra
```

```
plot,two,teen,couple,go,church,party,drink,drive,get,accident,one,guy,die,girlfriend,continue,see,life,nightm
happy,bastard,quick,movie,review,damn,y2k,bug,get,head,start,movie,star,jamie,lee,curtis,another,baldwin,brot
movies,like,make,jade,movie,viewer,thankful,invention,timex,indiglo,watch,base,late,1960,television,show,name
quest,camelot,warner,bros,first,feature,length,fully,animate,attempt,steal,clout,disney,cartoon,empire,mouse,
synopsis,mentally,unstable,man,undergo,psychotherapy,save,boy,potentially,fatal,accident,fall,love,boy,mother
capsule,2176,planet,mar,police,take,custody,accuse,murderer,face,title,menace,lot,fight,whole,lot,story,other
ask,8mm,eight,millimeter,really,wholesome,surveillance,man,lose,sight,value,become,enmesh,seedy,sleazy,underw
exactly,long,movie,felt,even,nine,laugh,nine,months,terrible,mess,movie,star,terrible,mess,man,mr,hugh,grant,
```

Noun Lemmatization / Verb Lemmatization

In this example there are 2 situations. First, there is the plural/single problem. Secondly, there is the '-ing' problem.

First problem, in verb lemmatized words. There are *movies* and *movie* words. They are the same words, just one of them is plural. So, we need to remove this '-s'.

Second problem, in noun lemmatized words. There is *dy*. It's meaningless. So,we need to convert to something meaningful.

Because of these problems we use both lemmatization methods.

```
plot,two,teen,couple,go,church,party,drink,drive,get,accident,one,guy,die,girlfriend,continue,see,life,nightr
happy,bastard,quick,movie,review,damn,y2k,bug,get,head,start,movie,star,jamie,lee,curtis,another,baldwin,bro
movies,like,make,jade,movie,viewer,thankful,invention,timex,indiglo,watch,base,late,1960,television,show,name
quest,camelot,warner,bros,first,feature,length,fully,animate,attempt,steal,clout,disney,cartoon,empire,mouse,
synopsis,mentally,unstable,man,undergo,psychotherapy,save,boy,potentially,fatal,accident,fall,love,boy,mothe
capsule,2176,planet,mar,police,take,custody,accuse,murderer,face,title,menace,lot,fight,whole,lot,story,othe
ask,8mm,eight,millimeter,really,wholesome,surveillance,man,lose,sight,value,become,enmesh,seedy,sleazy,underv
exactly,long,movie,felt,even,nine,laugh,nine,months,terrible,mess,movie,star,terrible,mess,man,mr,hugh,grant,
```

```
plot,two,teen,couple,go,church,party,drink,drive,get,accident,one,guy,die,girlfriend,continue,see,life,nightm
happy,bastard,quick,movie,review,damn,y2k,bug,get,head,start,movie,star,jamie,lee,curtis,another,baldwin,brot
movie,like,make,jade,movie,viewer,thankful,invention,timex,indiglo,watch,base,late,1960,television,show,name,
quest,camelot,warner,bros,first,feature,length,fully,animate,attempt,steal,clout,disney,cartoon,empire,mouse,
synopsis,mentally,unstable,man,undergo,psychotherapy,save,boy,potentially,fatal,accident,fall,love,boy,mother
capsule,2176,planet,mar,police,take,custody,accuse,murderer,face,title,menace,lot,fight,whole,lot,story,other
ask,8mm,eight,millimeter,really,wholesome,surveillance,man,lose,sight,value,become,enmesh,seedy,sleazy,underw
exactly,long,movie,felt,even,nine,laugh,nine,month,terrible,mess,movie,star,terrible,mess,man,mr,hugh,grant,h
```

Noun Lemmatization / Double Method Lemmatization

As you can see we fix the problems. *movies* turned to *movie*, *dy* turned to *die*.

B.1 Different General Features

- Explain why you choose these features in detail, why you think that these features would be proper for the data?

In this project, we choose 4 different general features. All features result from visual inspection of data.

- Longest words in text files
- Most used words in text files
- 4 digits year count in text files
- Subjectivity of data

Longest words and most used words are general features for text data. On visual inspection of data, we realize there are dates. So we decided to find them. Finally, some text has subjectivity words. We find them also.

B.2 BoW vs TF-IDF

- Explain the comparison of these two models in detail, the one that you choose and why you think that it is more suitable for the data?

In this project, we tried these two models. Firstly, we thought of using bags of words. There is a result of bags of words.

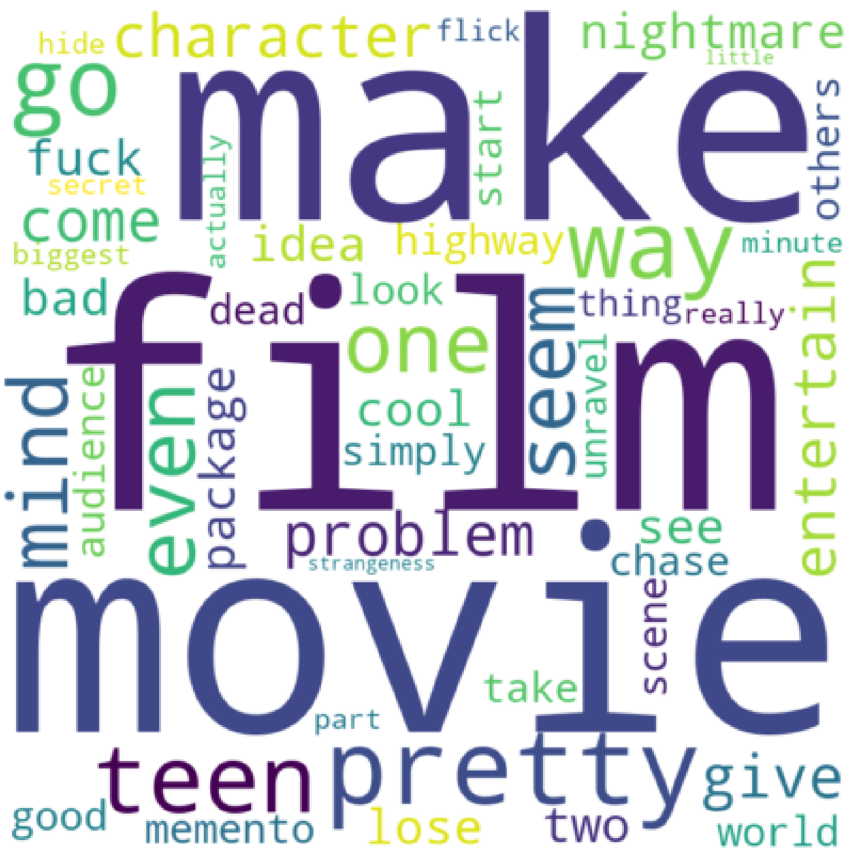
File Index	character	film	get	go	like	make	movie	one	see	time
0	3	8	4	4	3	7	7	3	2	0
1	0	0	2	2	4	0	5	0	0	1
2	3	9	1	2	4	2	3	5	2	2
3	1	1	1	2	1	0	0	2	2	1
4	1	3	1	0	1	0	3	3	1	3
5	0	11	0	1	2	5	2	0	2	3
6	1	7	1	1	4	2	1	0	3	2
7	2	0	1	1	1	0	7	3	0	1
8	2	4	1	1	0	2	3	1	0	4
9	0	9	1	2	0	1	8	3	5	3

When we passed to the TF-IDF model, the result is below.

File Index	character	film	get	go	like	make	movie	one	see	time
0	0.221058	0.488749	0.289414	0.302268	0.209775	0.471504	0.461296	0.184482	0.154863	0.000000
1	0.000000	0.000000	0.297824	0.311051	0.575656	0.000000	0.678144	0.000000	0.000000	0.152610
2	0.273088	0.679259	0.089383	0.186706	0.345534	0.166424	0.244231	0.379839	0.191313	0.183206
3	0.250200	0.207443	0.245676	0.513174	0.237430	0.000000	0.000000	0.417605	0.525837	0.251776
4	0.174778	0.434730	0.171618	0.000000	0.165858	0.000000	0.468927	0.437579	0.183663	0.527638
5	0.000000	0.814263	0.000000	0.091560	0.169449	0.408070	0.159694	0.000000	0.187640	0.269532
6	0.120178	0.697483	0.118005	0.123246	0.456177	0.219714	0.107479	0.000000	0.378861	0.241870
7	0.273719	0.000000	0.134385	0.140353	0.129875	0.000000	0.856784	0.342646	0.000000	0.137722
8	0.299967	0.497410	0.147272	0.153812	0.000000	0.274206	0.402404	0.125168	0.000000	0.603714
9	0.000000	0.597695	0.078650	0.164287	0.000000	0.073220	0.573078	0.200537	0.420853	0.241810

When we look at these results, we decide to use the TF-IDF model. Because, in the TF-IDF model there is frequency of a word. If we compare two files with BoW, we cannot say ‘This word is more used than the other file.’ In the TF-IDF model we can surely.

B.3 Word Clouds



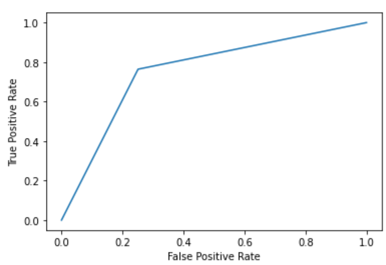
As we can see, there are the most used words: film, movie and make.

B.4 Machine Learning Algorithms

- Explain why you choose these classification and tree based models in detail and compare 4 machine learning algorithms that you used in terms of performance measures.

  In this project, we decided to use **XGBoost** for the tree based method and **Linear Regression** for the classification method. When we look at the result of tree based methods in other text mining process projects, XGBoost has a significant superiority to other methods.[1]

```
XGBoost
Accuracy : %75.65
Precision: 0.75
F1 Score : 0.76
Root Mean Squared Error : 0.49
```

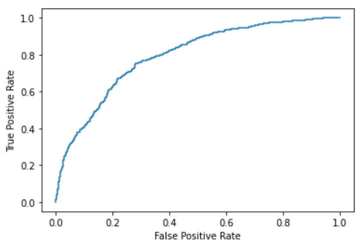


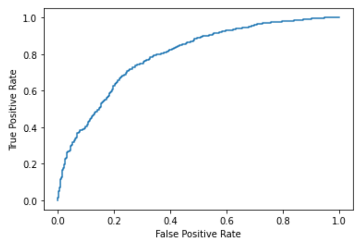
predicted	0	1
Target		
0	749	251
1	236	764

For classification method choosing, we compare **Logistic Regression** with **Linear Regression**.

```
Linear Regression
Accuracy : %73.15
Precision: 0.72
F1 Score : 0.74
Root Mean Squared Error : 0.52
```

```
Logistic Regression
Accuracy : %72.60
Precision: 0.72
F1 Score : 0.73
Root Mean Squared Error : 0.52
```



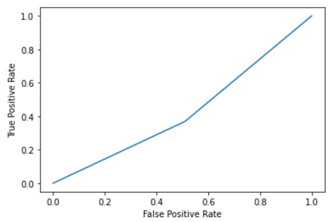


predicted	0	1
Target		
0	705	295
1	242	758

predicted	0	1
Target		
0	706	294
1	254	746

We tried to use the K-Means algorithm but its result is not very good.

```
K-Means
Accuracy : %42.95
Precision: 0.42
F1 Score : 0.39
Root Mean Squared Error : 0.76
```



predicted	0	1
Target		
0	490	510
1	631	369

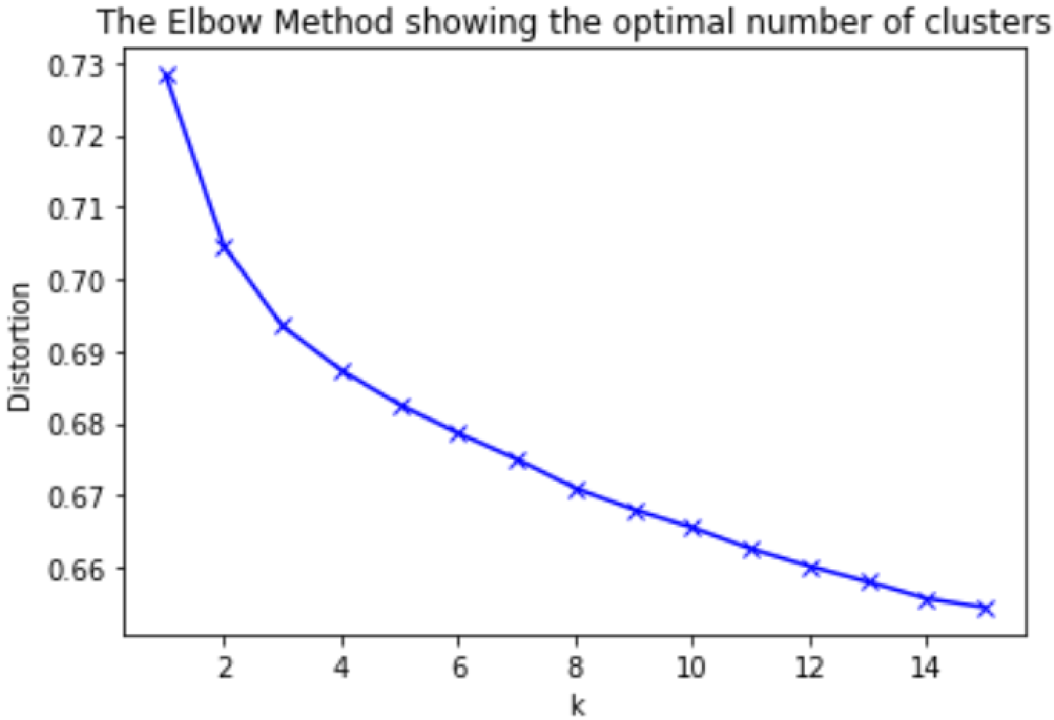
C. Topic Modelling with LDA and K-Means

In section B4, we used the elbow method to find the value of k. Although the values we had in mind were 2, we did not have any target value that we could compare in the data. Therefore, we labeled this data, separated as negative and positive, according to these two values.

In section E, we discussed the relevance values we obtained with LDA while finding the most suitable number of titles. We assumed that if the relevance value is greater than 0.1, it has the potential to become a title. And there were 2 values corresponding to this value.

If the data were properly labeled, we could use the k value we found with the elbow method. In this way, the number we obtained with the number of headers and the number we obtained with the elbow method would be almost the same.

Topic	Relevance
43	0.477665
32	0.125897
86	0.061152
82	0.047508
57	0.025856
29	0.021423
95	0.020913
78	0.017571
35	0.015155
81	0.015002



E. Sentiment Analysis

First, we cross-validated the simplest logistic regression model with TF-IDF and then with BoW(Bag of Words). According to the resulting score values, TF-IDF showed us that it is a more successful algorithm. Then, based on our previous experiences, we passed the RFC (Random Forest Classifier) and DTC (Decision Tree Classifier) models through the cross validation algorithm again. We decided that the most suitable of the 3 models for our data was Logistic Regression. Because this was the model with the highest accuracy rate. Then we tried all possibilities with grid search to determine the best hyperparameters of the model we chose. And surprisingly, we got a better result in test data than train data.
If we had huge big data, maybe the RFC would have been more successful than the model we specified.

```
Log. Reg. TFIDF:  0.8013333333333333
Log. Reg. BoW:  0.6426666666666667
Random Forest TFIDF:  0.768
DTC TFIDF:  0.6266666666666667
Best Estimator =  Pipeline(steps=[('vect', TfidfVectorizer(lowercase=False)),
                ('log_reg', LogisticRegression(C=200, random_state=0))])
Score=  0.83
The model has a test accuracy of 85.60%
```

F. References

[1]www.towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d