Sentiment analysis experiments

Dr. Eghbal mansoori[a,c,1] **and Nima Iji**[b,1,2]

[a]University of Shiraz; [b]University of Shiraz

This manuscript was compiled on September 12, 2021

Natural Language Processing | Machine Learning | Classification | Clustering

1. Introduction

This experimental project is all about, how we can measure each pair of two sentences similarity? how text vectorization helps us to do arithmetic operations on sentences?

The project has two experiments that shown in Fig1 diagram, one of them uses "The introductions dataset" and finds out clusters in gathered documents. Second one, is about the news classification that creates a machine learning model by NLP methods to classifies news, into 40 different categories.

2. Datasets

A. The introductions dataset. This dataset contains 12 different documents in .txt format that accumulated from papers' introductions, and websites articles about "Linear regression" method manually.

B. The news dataset. This dataset contains around 200k news headlines from the year 2012 to 2018 obtained from HuffPost. The model trained on this dataset could be used to identify tags for untracked news articles or to identify the type of language used in different news articles. You can access this database from Kaggle.

3. Project hierarchy 1

A. Corpus loader. Both experiments start by using the "CorpusLoader" class, which is hired to maintain our dataset (unstructured data) and organize instances into ordered and clear data structures. The CorpusLoader gets a path from the programmer, and depends on files formats (.txt or .json) loads a dataset into a single Pandas DataFrame class and a NumPy array, and both are accessible by inner methods.

B. Text normalization. In this step, we are going to clean the dataset from vain vocabularies like, stop words, punctuation ,or numbers; These vocabularies have no influence on the outputs and just have cost enormous of useless operations. we recognize these vocabularies by three methods, *isStopword, isPunct or isNumber*, and clean our instances. In addition, we use ***NLTK lemmatization*** to reformat each word to its root depends on its context.

C. Vectorization. After we cleaned the dataset, it's ready to vectorize instances. it goes without saying that, the second experiment uses three different vectorizations (Count Vectorizer, TFIDF Vectorizer, and Hashing Vectorizer) to find the best accuracy score of the classification. All vectorizer functions come from the *SciKit-Learn* python library, to speed up the development.

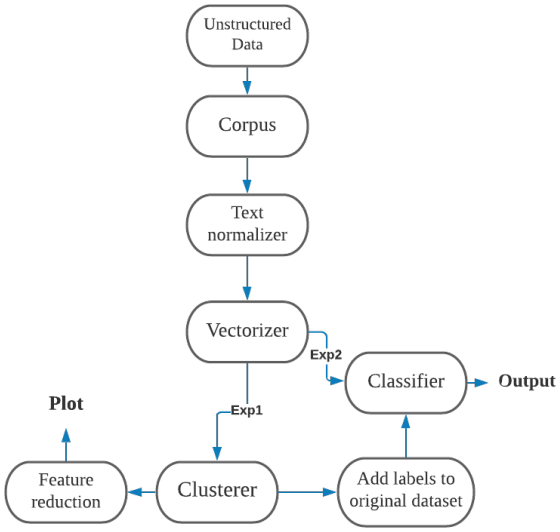

Fig. 1. Vertical workflow of experiments.

D. Machine learning models. In the end, we can create ML models with the prepared dataset. For the unsupervised dataset, we provide a clustering model that labels each instance. On the other hand, we use a classification model to classify the labeled dataset.

Table 1. Comparison of the Silhouette scores

K	Average score
2	**0.66143**
3	0.14293
4	0.30976
5	0.14484
6	0.11842

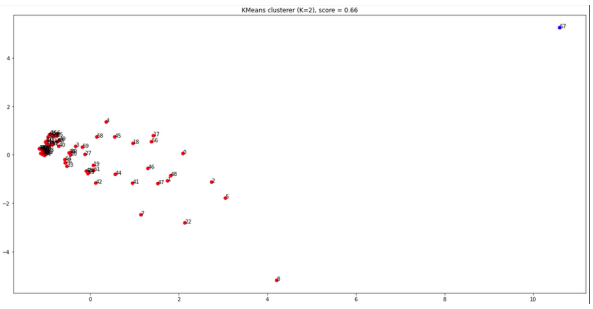

Fig. 2. Clusters in after using PCA method.

4. Experiments

A. Clustering and classifying The introductions dataset. First of all, we load and preprocess the dataset by using the CorpusLoader and the TextNormalizer; Then we use the MiniBatchKMeans tool from the SciKit-Learn library to cluster the dataset with low accuracy and high speed for week resources.

Despite we repeat this process for different numbers of clusters and check their Silhouette score to select the best score for labeling the dataset.

To visualize the clusters, we use the PCA dimension reduction method to transfer instances in two dimensions.2 In the end, we use a classification algorithm (Stochastic gradient descent) for the measurement of the clusters. You can overlook the classification report in Table 2.

B. News categorization. The goal of this experiment is to categorize news by processing their sentences. Nevertheless, we load the dataset by the CorpusLoader and drop authors, link, and date fields, because they are not recognizable patterns in topic sentences; then we normalize the headlines and short descriptions fields and encode the category field for machine learning algorithms.

Table 2. Classification report for Exp. 1

Label	precision	recall	f1-score	support
1	1.00	0.42	0.59	12
2	0.67	1.00	0.80	2
3	0.00	0.00	0.00	0
4	0.33	1.00	0.50	1
5	0.00	0.42	0.00	0
accuracy			0.53	15
macro avg	0.40	0.48	0.38	15
weighted avg	0.91	0.53	0.61	15

For better validation and model training, we use the cross-validation method for seven splits. The next step is to select a text vectorizer for the best accuracy score; You can the results in Tables3 - 6.

Table 3. Classifying headlines vectors by count vectorizer and SGD-Classifyer.

Split	Accuracy score
1	0.4842
2	0.4781
3	0.4808
4	0.4845
5	0.4776
6	0.4808
7	**0.4815**

Table 4. Classifying short descriptions vectors by count vectorizer and SGDClassifyer.

Split	Accuracy score
1	0.3546
2	0.3671
3	0.3653
4	0.3689
5	0.3630
6	0.3728
7	**0.3610**

Table 5. Classifying headlines vectors by TFIDF vectorizer and SGD-Classifyer.

Split	Accuracy score
1	0.5396
2	0.5453
3	0.5471
4	0.5405
5	0.5442
6	0.5478
7	**0.5420**

Table 6. Classifying headlines vectors by Hashing vectorizer and SGDClassifyer.

Split	Accuracy score
1	0.5371
2	0.5373
3	0.5365
4	0.5389
5	0.5360
6	0.5400
7	**0.5425**

B.1. Concrete results. We can find out short descriptions are not cohesive and have no recognizable pattern. So we produce a

preliminary result that short descriptions are not a proper parameter for predicting news categories. In contrast, headlines sentences are a good feature for predicting news categories.

Another result is that TFIDF and Hashing vectorizer doesn't influence outputs results and the huge difference is the data loading pace; Hashing vectorizers is a useable tool for vectorizing big scale text datasets and we can see that there is no big difference with the presence of hashing collisions.