Report for Assignment

Mahdieh Sajedi Pour

10/24/2023

1) Overview and Problem Statement

This project aims to teach us some fundamental and important techniques in NLP. We were supposed to use different vectorizers and classifiers (listed below) to do a Persian sentiment analysis task on the SnappFood dataset

a) Vectorizers and embedding techniques:
   i) CountVectorizer(sklearn)
   ii) TfidfVectorizer(sklearn)
   iii) Word2Vec(gensim)
   iv) FastText(gensim)

b) Classifiers:
   i) LogisticRegression(sklearn)
   ii) MultinomialNB(sklearn)
   iii) RandomForestClassifier(sklearn)

2) Data Preparation

The Dataset consists of 70000 comments(35000 positive and 35000 negative) from an online food delivery company in Iran.

a) WordTokenizer

Each comment needs to be tokenized using a word tokenizer so that we can perform other preprocessing tasks. It splits each data item into words but keeps two-part verbs as a whole. In this step, we also replace numbers and IDs.

```
0    [واقعا, حیف, وقت, که, بنویسم, سرویس, دهیتون, ش...
1    [ساعته, برسه, ولی, نیم, ساع, 1, NUM, قرار, بود...
2    [قیمت, این, مدل, اصلا, با, کیفیتش, سازگاری, ند...
3    [عالی, بود, همه, چه, درست, و, به, اندازه, و, ک...
4                 [شیرینی, وانیلی, فقط, یک, مدل, بود, .]
Name: comment, dtype: object
```

b) Stemmer

Stemming tries to reduce words to their base form (sometimes it produces meaningless words). We use it to make the data easier to process.

```
0    [واقعا, حیف, وق, که, بنویس, سرویس, دهیتون, شده...
1    [ساعته, برسه, ول, ن, ساع, ز, 1, NUM, قرار, بود...
2    [قیم, این, مدل, اصلا, با, کیفیت, سازگار, نداره...
3    [عال, بود, همه, چه, درس, و, به, اندازه, و, کیف...
4                    [شیرین, وانیل, فقط, یک, مدل, بود, .]
Name: comment, dtype: object
```

c) Lemmatizer

Lemmatization is also a way to reduce a word to its base form, but unlike stemming, it takes into account the context of the word, and it produces a valid word.

```
0    [واقعا, حیف, وق, که, بنویس, سرویس, دهیتون, شده...
1    [ساعته, برسه, ول, #هست, 1, NUM, قرار, بود#باش...
2    [قیم, این, مدل, اصلا, با, کیفیت, سازگار, نداره...
3    [عال, بود#باش, همه, چه, درس, و, به, اندازه, و...
4                 [شیرین, وانیل, فقط, یک, مدل, بود#باش, .]
Name: comment, dtype: object
```

d) Last steps

Here we remove Persian stop words, join the words in each comment again, and convert the dataframe into a list. Then, we use train_test_split with test_size=0.15.

```
['واقعا حیف وق بنویس سرویس دهیتون افتضاح',
 'ساعته برسه ول #هست ساع زود موقع ، دید#بین چقدر پلاک خفنهه ، سالهاس مشتریشون سالهاس مزه میده غذاشون 1 NUM قرار بود#باش',
 'قیم مدل اصلا کیفیت سازگار نداره ، ظاهر فریبنده داره ، میکنن کالباس قارچ',
 'عال بود#باش درس اندازه کیف ، امیداور کیفیتون باشه مشتر همیشگ بش',
 'شیرین وانیل مدل بود#باش .']
```

3) Methodology

Due to the fact that ML models can not process texts as we do, we need to provide an understandable representation for our models.

i) CountVectorizer(sklearn)

This is the simplest method to convert the comments into vectors. It shows the presence of a word in the comment with 1 and its absence with 0. Here is the vectorized form of train data:

	11	aa	aali	aallii	ab	ablimo	about	acting	adasi	adasish	...	یکیف	یکیلو	یکیه	یکیو	یگ	یگانه	یگیر	ییب	ییسکو	ییسکوی
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59053	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59054	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59055	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59056	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
59057	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

ii) TfidfVectorizer(sklearn)

TF-IDF stands for Term Frequency-Inverse Document Frequency.

Term frequency shows the number of times that a word is repeated in a comment divided by the total number of unique words in all comments.

Inverse data frequency is equal to the log of the total number of comments divided by the number of comments that contain the word. 1 may be included for standardization.

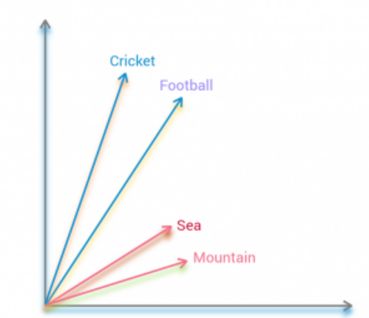
- $\mathrm{IDF}(t) = \log\frac{1+n}{1+\mathrm{df}(t)} + 1$

Then, by multiplying TF with IDF we have the final result.

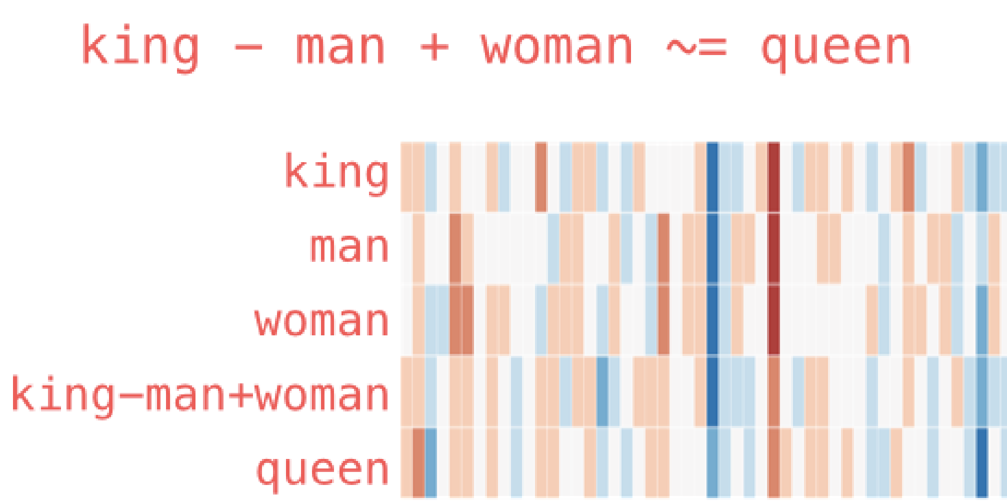
	11	aa	aaaallliiii	aali	aalllii	ab	ablimo	acting	adam	adasi	...	یکیلو	یکیه	یکیو	یگ	یگانه	یگیر	یی	ییب	ییسکو	ییسکوئ
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
59053	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
59054	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
59055	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
59056	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
59057	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

iii) Word2Vec(gensim)

Word2Vec is introduced so that two words with similar contexts and similar meanings will have a similar vector representation. For example, “computer”, “keyboard,” and “mouse” are frequently used in similar contexts thus they have a similar vector representation in a Word2Vec model. Another example:



Here we use the fact that the meaning of a word can be inferred by its surrounding words. By analyzing n-grams, we are able to do such a thing and gain insights into the relationships between words or in a given text. There is a common example that is color code (red if they’re close to 2, white if they’re close to 0, blue if they’re close to -5) shown below:

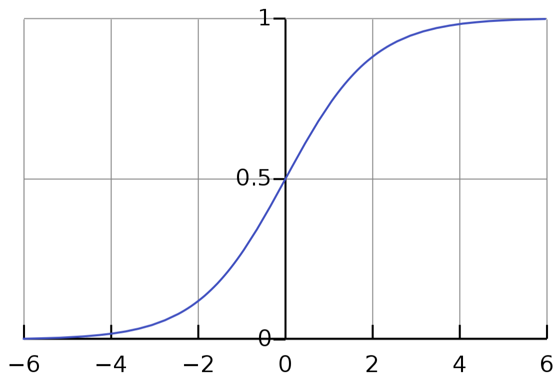


The result of the operation "king-man+woman" is not exactly equal to "queen", but "queen" is the closest present word from the embedding list.

iv) FastText(gensim)
Unlike the word2vec model that provides embedding to the words, fastText provides embeddings to the character n-grams. The plus point of FastText is that it can also find the word embeddings that are not present in the training phase (word2vec cannot!). In this technique, each word is represented as the average of the vector representation of its character n-grams along with the word itself. For example, consider the word “present” and n = 3, then the word will be represented by character n-grams:
< pr, pre, res, ese, sen, ent, nt > and < present>
So, the word embedding for the word present is given as the sum of all vector representations of all of its character n-gram and the word itself.

Then, it is time to classify the embeddings we have. There are many ways and classifiers that we can use.

v) LogisticRegression(sklearn)
It is a statistical method that uses a logistic function to model the relationship between the input features(word embeddings) and the output class(Happy or Sad). The logistic function maps any input value to a value between 0 and 1, which is the probability of the input belonging to a certain class.



The algorithm’s aim is to find the best parameters that minimize the difference between the predicted value and the real class labels in the training data. When the best parameters are found, they can be used to predict the label of an unseen data item.

vi) MultinomialNB(sklearn)
Each feature has a probability of being seen in a class. The algorithm calculates the probability of each feature given each class and then multiplies these probabilities together to get the total probability of a comment belonging to a particular class. The class with the highest probability is considered as the label of the input.

vii) RandomForestClassifier(sklearn)
This method combines the output of multiple decision trees to reach a single result. In this process, a comment and a subset of features are used to construct each decision tree. Individual decision trees are built for each sample. Then, each decision tree will generate an output. The final output is based on the majority.

4) Result

We tried each embedding method with different classifiers and the results are shown below:

i) LogisticRegression

(1) CountVectorizer

	precision	recall	f1-score	support
0	0.85	0.82	0.83	5275
1	0.82	0.85	0.83	5147
accuracy			0.83	10422
macro avg	0.83	0.83	0.83	10422
weighted avg	0.83	0.83	0.83	10422

(2) TfidfVectorizer

	precision	recall	f1-score	support
0	0.87	0.80	0.84	5275
1	0.81	0.88	0.84	5147
accuracy			0.84	10422
macro avg	0.84	0.84	0.84	10422
weighted avg	0.84	0.84	0.84	10422

(3) word2vec

	precision	recall	f1-score	support
0	0.72	0.65	0.68	5275
1	0.67	0.75	0.71	5147
accuracy			0.70	10422
macro avg	0.70	0.70	0.70	10422
weighted avg	0.70	0.70	0.70	10422

(4) FastText

	precision	recall	f1-score	support
0	0.72	0.64	0.68	5275
1	0.67	0.75	0.71	5147
accuracy			0.70	10422
macro avg	0.70	0.70	0.69	10422
weighted avg	0.70	0.70	0.69	10422

ii) MultinomialNB

(1) CountVectorizer

	precision	recall	f1-score	support
0	0.87	0.78	0.82	5275
1	0.80	0.88	0.84	5147
accuracy			0.83	10422
macro avg	0.83	0.83	0.83	10422
weighted avg	0.83	0.83	0.83	10422

(2) TfidfVectorizer

	precision	recall	f1-score	support
0	0.88	0.75	0.81	5275
1	0.78	0.89	0.83	5147
accuracy			0.82	10422
macro avg	0.83	0.82	0.82	10422
weighted avg	0.83	0.82	0.82	10422

(3) word2vec

	precision	recall	f1-score	support
0	0.61	0.69	0.65	5275
1	0.63	0.54	0.59	5147
accuracy			0.62	10422
macro avg	0.62	0.62	0.62	10422
weighted avg	0.62	0.62	0.62	10422

(4) FastText

	precision	recall	f1-score	support
0	0.60	0.68	0.64	5275
1	0.62	0.53	0.58	5147
accuracy			0.61	10422
macro avg	0.61	0.61	0.61	10422
weighted avg	0.61	0.61	0.61	10422

iii) RandomForestClassifier

(1) CountVectorizer

	precision	recall	f1-score	support
0	0.86	0.80	0.83	5275
1	0.81	0.86	0.84	5147
accuracy			0.83	10422
macro avg	0.83	0.83	0.83	10422
weighted avg	0.83	0.83	0.83	10422

(2) TfidfVectorizer

	precision	recall	f1-score	support
0	0.87	0.79	0.83	5275
1	0.80	0.88	0.84	5147
accuracy			0.83	10422
macro avg	0.83	0.83	0.83	10422
weighted avg	0.83	0.83	0.83	10422

(3) word2vec

	precision	recall	f1-score	support
0	0.73	0.67	0.70	5275
1	0.69	0.74	0.72	5147
accuracy			0.71	10422
macro avg	0.71	0.71	0.71	10422
weighted avg	0.71	0.71	0.71	10422

(4) FastText

	precision	recall	f1-score	support
0	0.73	0.67	0.70	5275
1	0.69	0.75	0.72	5147
accuracy			0.71	10422
macro avg	0.71	0.71	0.71	10422
weighted avg	0.71	0.71	0.71	10422

As can be seen, word2vec and fasttext methods are not suitable for our task. The reason is that our dataset is not big enough. The other two methods perform approximately the same. Regarding the classifying methods, LogisticRegression gives the best accuracy when it is fed with TfIdf vectors.