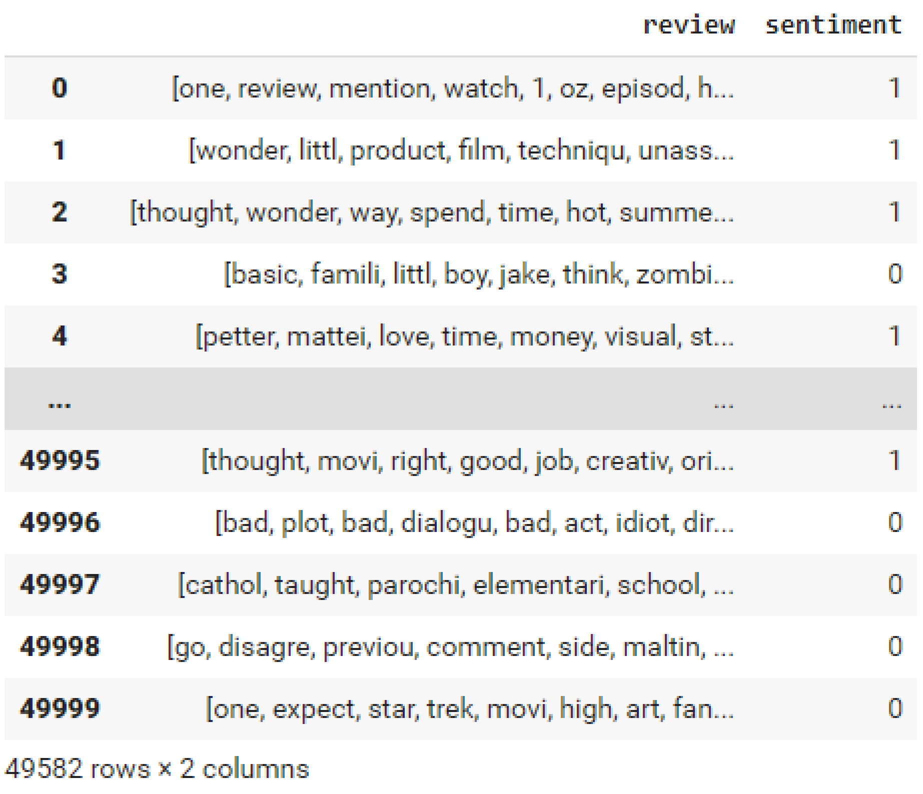
Homework 2: Text Analysis and Classification

<u>Question 1 - Text pre-processing and exploration:</u>

First, I **loaded** the IMDB dataset into 'df' variable. Then I made a **pre-processing** phase which contains the cleaning and normalizing of the text. The pre-processing steps are:

1. Drop duplicate reviews.
2. Remove html tags like <br>
3. Change all characters to lower case.
4. Remove special characters like ?,!, etc.
5. Tokenization.
6. Remove stop words.
7. Stemming
8. Label encoding – positive =1, negative = 0

The dataset after this step:

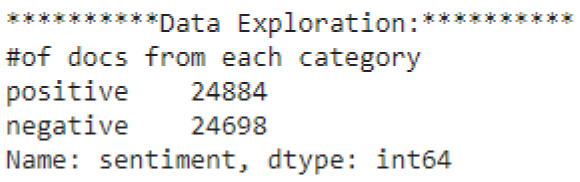


	review	sentiment
0	[one, review, mention, watch, 1, oz, episod, h...	1
1	[wonder, littl, product, film, techniqu, unass...	1
2	[thought, wonder, way, spend, time, hot, summe...	1
3	[basic, famili, littl, boy, jake, think, zombi...	0
4	[petter, mattei, love, time, money, visual, st...	1
...	...	...
49995	[thought, movi, right, good, job, creativ, ori...	1
49996	[bad, plot, bad, dialogu, bad, act, idiot, dir...	0
49997	[cathol, taught, parochi, elementari, school, ...	0
49998	[go, disagre, previou, comment, side, maltin, ...	0
49999	[one, expect, star, trek, movi, high, art, fan...	0

49582 rows × 2 columns

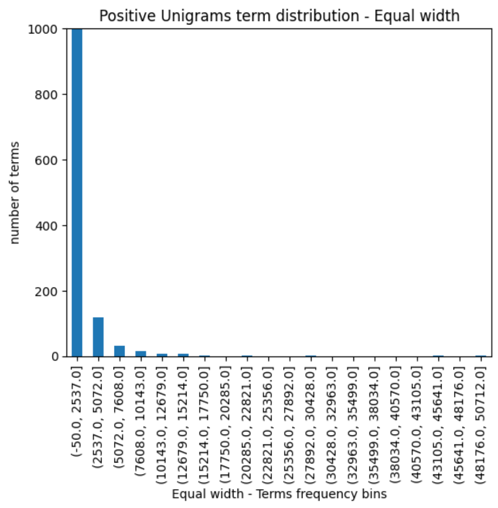
Next, I made the **Data exploration phase**.

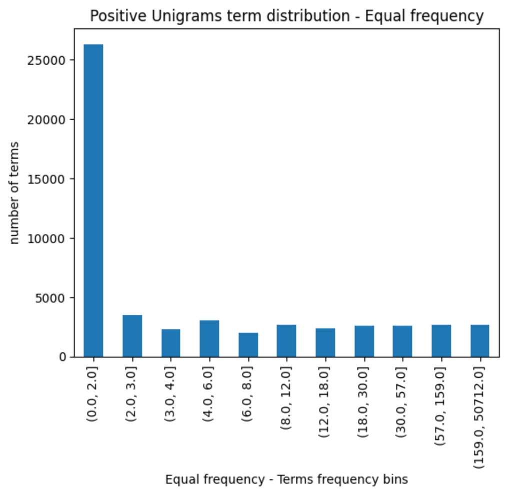
- Number of docs from each category (there are less than 25,000 reviews because of the duplicates removal):

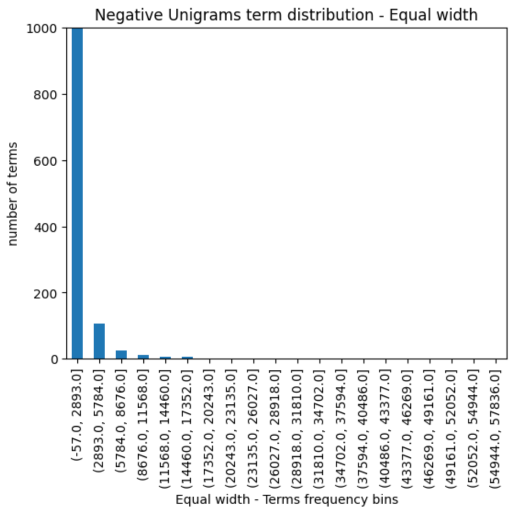


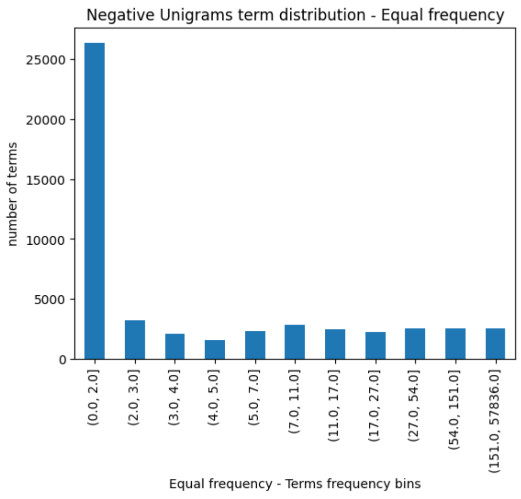
```
**********Data Exploration:**********
#of docs from each category
positive    24884
negative    24698
Name: sentiment, dtype: int64
```

- Terms distribution per category. This phase was divided to unigrams and bigrams. First, I created a list of all the words of each category. Then, I used a Counter object in order to count how many times each word exists on each category. And then I divided the terms into frequency bins, in which each bin contains the terms that their frequency falls into that bin. The bins were chosen by two metrics – equal depth which each bin has equal range, and equal frequency in which each bin has approximately the same number of terms. The Unigram graphs I got:

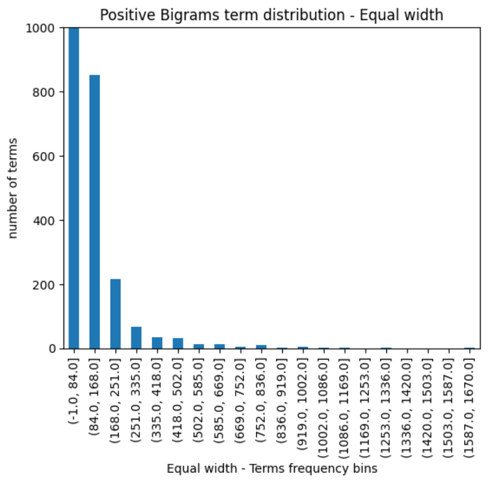


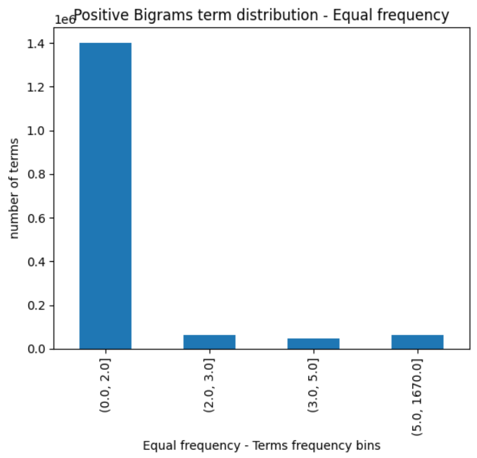


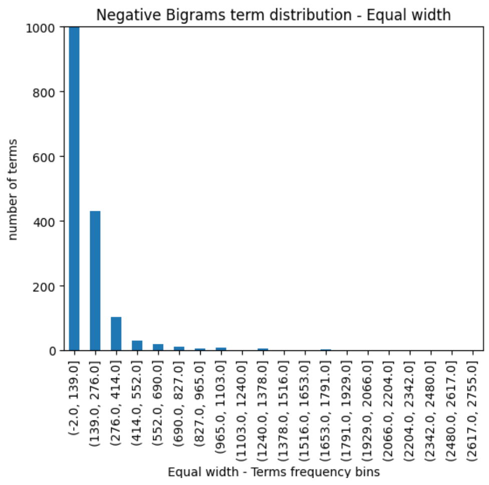


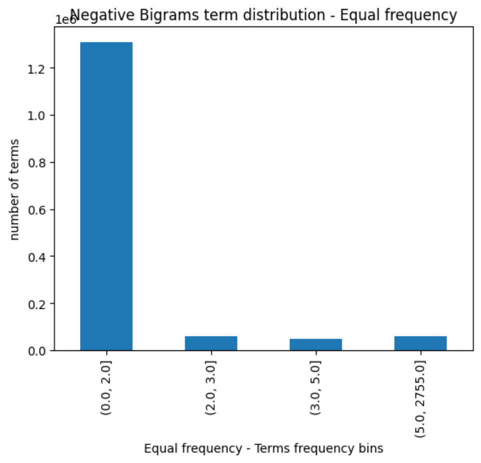


Bigrams graphs:



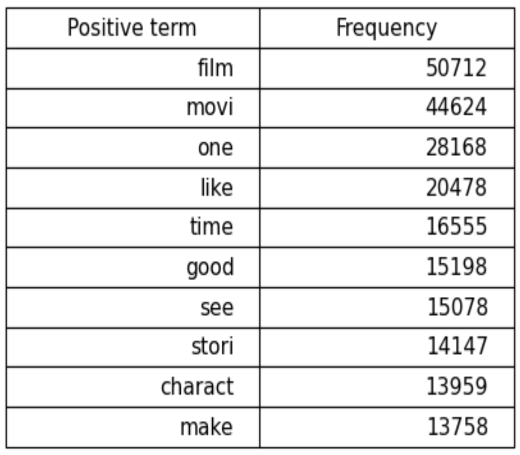






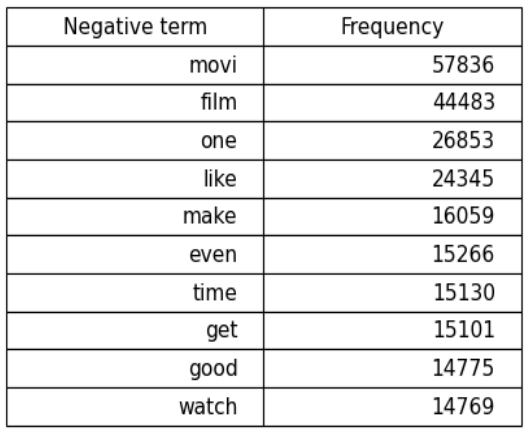
- In order to present the table of top 10 terms per category, I merged between the unigram and bigrams terms, and counted the times each word exists on each category and then used the Counter object function most_common(?) in order to get the top 10 terms.

Top 10 positive terms



Positive term	Frequency
film	50712
movi	44624
one	28168
like	20478
time	16555
good	15198
see	15078
stori	14147
charact	13959
make	13758

Top 10 negative terms



Negative term	Frequency
movi	57836
film	44483
one	26853
like	24345
make	16059
even	15266
time	15130
get	15101
good	14775
watch	14769

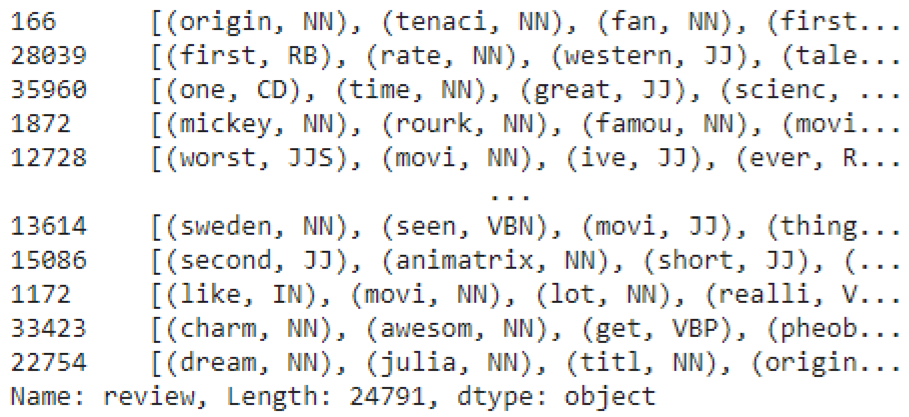
Finally, I split the data into 50% train set and 50% test sets. Before the splitting, I concatenated tokens into sentences in order to use feature extraction libraries.

Expected challenges:

- Same words can have different sentiment meaning - negation problem.
- Which words we should use - whether we keep only adjectives that may express opinions or all words.
- How to interpret features - bag of words, tf-idf, etc.
- As we can see from the exploration phase, the most common unigram words are very similar, but the bigrams have different meaning. For example, the term good exists in both classes.
- Most of the words exist few times on each class, so we got sparse vectors.

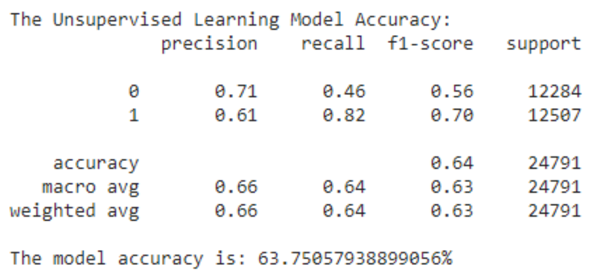
<u>Question 2 - Document classification – un-supervised learning:</u>

This phase used only the test set because I used pre-defined dictionary in the training phase. First, I used PerceptronTagger in order to set POS to every term. Example of the test set after the tagging:



```
166      [(origin, NN), (tenaci, NN), (fan, NN), (first...
28039    [(first, RB), (rate, NN), (western, JJ), (tale...
35960    [(one, CD), (time, NN), (great, JJ), (scienc, ...
1872     [(mickey, NN), (rourk, NN), (famou, NN), (movi...
12728    [(worst, JJS), (movi, NN), (ive, JJ), (ever, R...
                                ...
13614    [(sweden, NN), (seen, VBN), (movi, JJ), (thing...
15086    [(second, JJ), (animatrix, NN), (short, JJ), (...
1172     [(like, IN), (movi, NN), (lot, NN), (realli, V...
33423    [(charm, NN), (awesom, NN), (get, VBP), (pheob...
22754    [(dream, NN), (julia, NN), (titl, NN), (origin...
Name: review, Length: 24791, dtype: object
```

Then, I made sentiment analysis with sentiwordnet. Every term got a score – positive, neural, negative, so for each review, I summed the scores and calculated the average score. If the score was greater than 0 the review labeled as positive (1), otherwise the review labeled as negative (0). Finally, I calculated the accuracy of the model by comparing between the true labels and the predicted labels. The accuracy was 63.75%.



```
The Unsupervised Learning Model Accuracy:
              precision    recall  f1-score   support

           0       0.71      0.46      0.56     12284
           1       0.61      0.82      0.70     12507

    accuracy                           0.64     24791
   macro avg       0.66      0.64      0.63     24791
weighted avg       0.66      0.64      0.63     24791

The model accuracy is: 63.75057938899056%
```

<u>Question 3 - Document classification – supervised learning:</u>

The feature extraction methods I chose were: Bag of words and TF-IDF.

The machine learning methods I chose were: SVM, MultinomialNB, Perceptron.

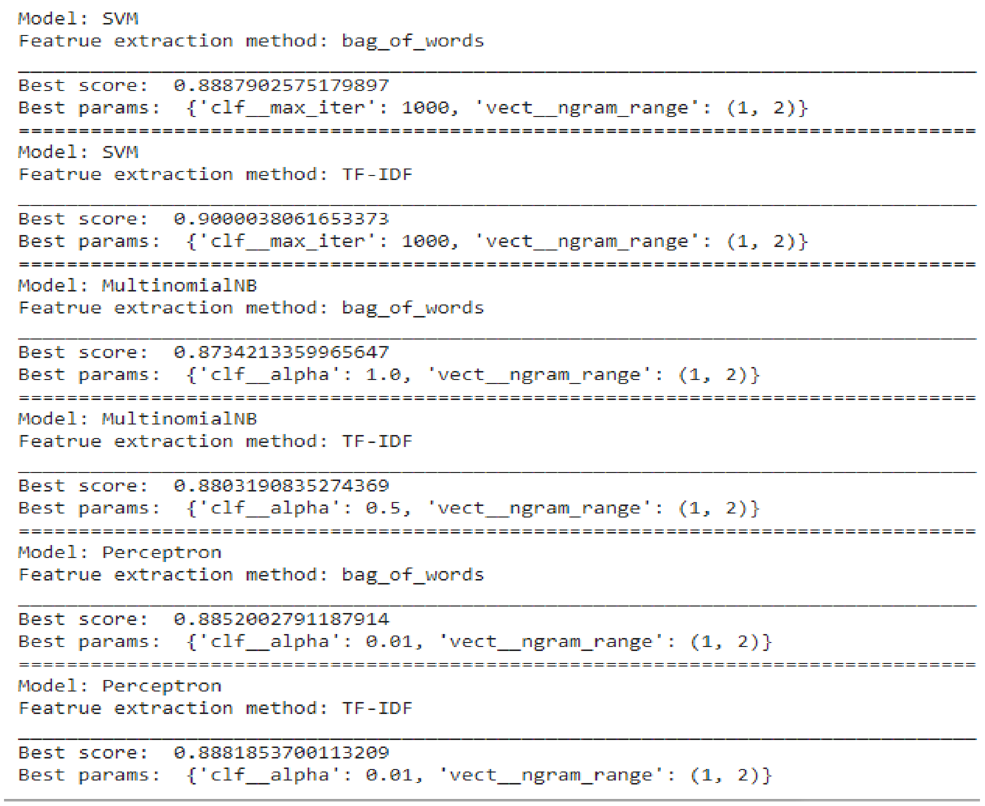
The parameter of the feature extraction method I have tested is ngram_range.

The parameters of the machine learning models I have tested were: for the SVM model I have tested the max_iter parameter, and for the other models I have tested the alpha parameter.

I used a Pipeline for each combination between feature extraction method and machine learning model.

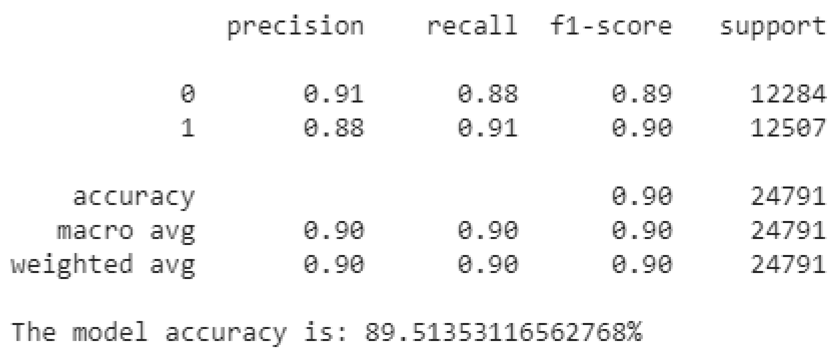
I also used Gridsearch approach in order to find the best parameters to use (tuning). It was using 10-fold cross validation.

The results I got for each combination are:



```
Model: SVM
Featrue extraction method: bag_of_words
________________________________________________________________________________
Best score:  0.8887902575179897
Best params:  {'clf__max_iter': 1000, 'vect__ngram_range': (1, 2)}
================================================================================
Model: SVM
Featrue extraction method: TF-IDF
________________________________________________________________________________
Best score:  0.9000038061653373
Best params:  {'clf__max_iter': 1000, 'vect__ngram_range': (1, 2)}
================================================================================
Model: MultinomialNB
Featrue extraction method: bag_of_words
________________________________________________________________________________
Best score:  0.8734213359965647
Best params:  {'clf__alpha': 1.0, 'vect__ngram_range': (1, 2)}
================================================================================
Model: MultinomialNB
Featrue extraction method: TF-IDF
________________________________________________________________________________
Best score:  0.8803190835274369
Best params:  {'clf__alpha': 0.5, 'vect__ngram_range': (1, 2)}
================================================================================
Model: Perceptron
Featrue extraction method: bag_of_words
________________________________________________________________________________
Best score:  0.8852002791187914
Best params:  {'clf__alpha': 0.01, 'vect__ngram_range': (1, 2)}
================================================================================
Model: Perceptron
Featrue extraction method: TF-IDF
________________________________________________________________________________
Best score:  0.8881853700113209
Best params:  {'clf__alpha': 0.01, 'vect__ngram_range': (1, 2)}
```

As we can see the best combination is: SVM + TF-IDF. So, I used this combination to predict the test set. The results are:



```
              precision    recall  f1-score   support

           0       0.91      0.88      0.89     12284
           1       0.88      0.91      0.90     12507

    accuracy                           0.90     24791
   macro avg       0.90      0.90      0.90     24791
weighted avg       0.90      0.90      0.90     24791

The model accuracy is: 89.51353116562768%
```

Task challenges and effective solutions

- All the challenges from the first question.
- Which machine learning model to use?
- Which feature extraction method to use?
- Which parameters to use? (classification model and feature extraction method)

Solutions –

- use grid search in order to find the best combination.
- In order to deal with sparsity - use sparse matrix.
- use bigrams in order to solve ambiguity.

<u>Question 4 - Supervised vs. Unsupervised learning approach</u>

Advantages & Disadvantages

The supervised learning model accuracy is 89.51% and the un-supervised learning model accuracy is 63.75%. The advantages of each approach:

Supervised –

- Much more accurate than unsupervised classification.
- No need to set Part of speech for each word.

Unsupervised –

- No prior knowledge is required; thus, the data is more objective.
- Can use words that does not exist in the train set.

Suggestion of an approach that combines both models:

Combined Approach Suggestion

1. Train the model with supervised learning approach.
2. predict the test set with the help of the train set
3. use the decision_function in order to see the confidence of each decision

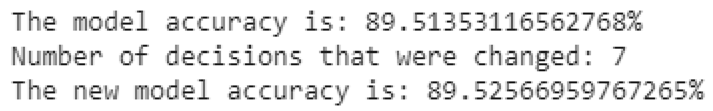
4. choose the decisions with low confidence
5. use the unsupervised learning approach to find out if according to this approach the decision is different
6. if the decision is different change the decision

Explanation:

Because the samples which their confidence is very low, we can guess that these samples contain neural words, or ambiguity words. So, if we use a pre-defined dictionary that may tag each word with its sentiment it maybe will change the decision of these samples.

Discussion

with this approach and with threshold of 0.0015, the decision of 7 samples was changed, and the accuracy increased a little by 0.012%.



```
The model accuracy is: 89.51353116562768%
Number of decisions that were changed: 7
The new model accuracy is: 89.52566959767265%
```