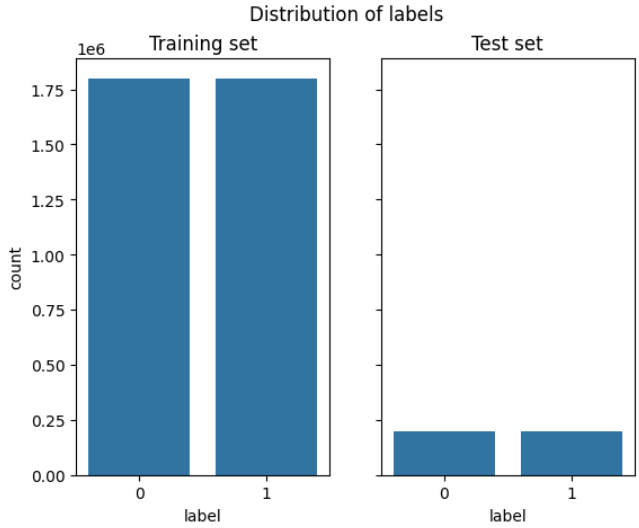
ML for NLP2 2023-2024

Lab 01 Feature-based classification

22210802 Haeeul HWANG

1. caching is a common strategy in software development to optimize performance, reduce dependencies on external servers, and enhance the user experience.

2.



The distribution of the labels is balanced, with an equal number of positive and negative reviews in the training and test set.
This is good because it means that the model will be trained on a balanced dataset and will be able to predict both positive and negative reviews.

3.

['best', 'book', 'don't', 'good', 'great']

There are 4 positive words among 5, 'don't' might come from the negative labels. So, most of the common words are positive ones.

4. Allowing users to split the data can introduce biases, especially if users unintentionally include data patterns that are specific to their use case.

Centralized splitting ensures a standardized and fair evaluation process across different models and research studies.

5.

```
# observation matrix using CountVectorizer

vectorizer = CountVectorizer(stop_words = 'english', max_features=1000, tokenizer=str.split)

X_train = vectorizer.fit_transform(dataset['train']['title'])
X_test = vectorizer.fit_transform(dataset['test']['title'])
```

```
print("Training set :", X_train.shape, "\n", X_train.toarray())
print("Test set :", X_test.shape, "\n", X_test.toarray())
```

✓ 5.4s

```
Training set : (3600000, 1000)
 [[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]]
Test set : (400000, 1000)
 [[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]]
```

6. Not always, we can just use ‘CountVectorizer’ to tokenize on this task.

7. Stop words are considered to be of little value in terms of conveying meaningful information (ex: "the," "and," "is," "in," and so on).
These words are very frequent and appear across various documents, but they typically do not contribute much to the content or meaning of the text.

8. It might depend on the tasks, but it would be efficient to use a pre-computed list of stop words.
For example, when I set the ‘max_df’ to 0.9 on this task, I still got some stop words such as ‘a’, ‘of’, ‘the’ as most common words.

9. ‘min_df’ stands for minimum document frequency. If we set the min_df value to 1, it means that certain word appears on only one document.
So, if we want to get more valuable information, we need the words which appears at least more than 2 documents.

10.
- min_df = 1

vocabulary size of training set : 540218

vocabulary size of test set : 127488

- min_df = 2

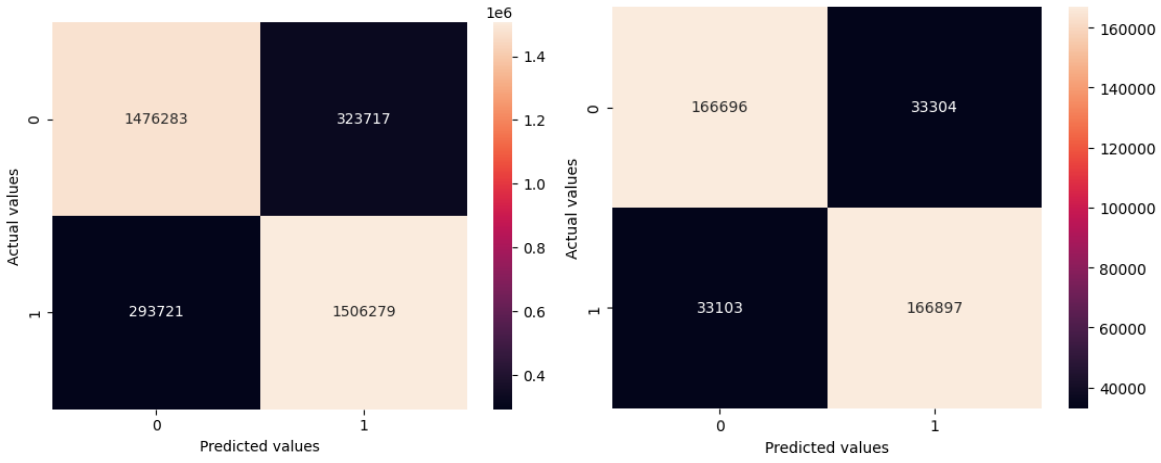
vocabulary size of training set : 182502

vocabulary size of test set : 44246

The size of vocabulary is smaller when we set ‘min_df’ to 2 and this means that there are fewer less important words.

11.

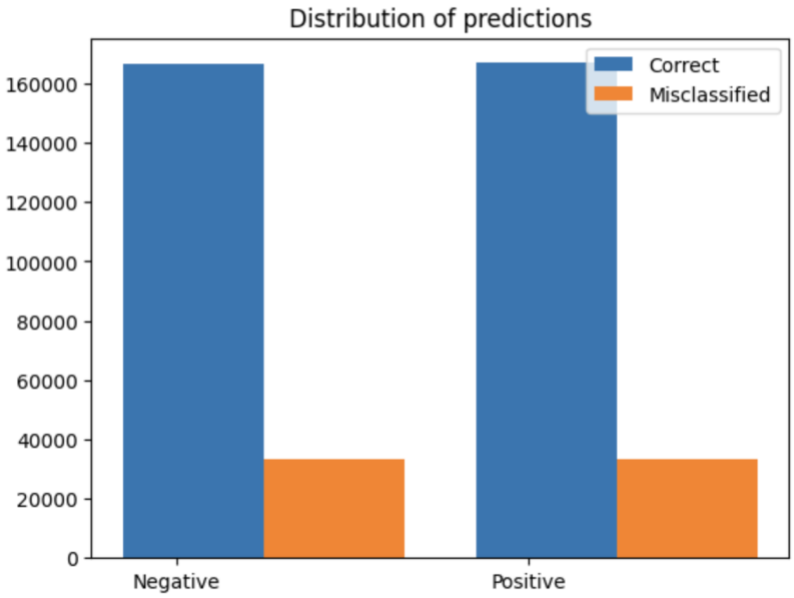
Accuracy on the training set: 0.828 Accuracy on the test set: 0.834



The accuracy is a little bit higher on the test set. (TP > TN > FP > FN)
The model is performing well in terms of identifying instances of the positive class.

12.

```
Correctly classified examples:  333593
Misclassified examples:  66407
```



13.

['australia!', 'outstanding!', 'fantastic!', 'rocks!', 'excellent!',
'jordan's', 'perfect!', 'l-8000', 'must-read', 'awesome!', 'excellant',
'fabulous!', 'wonderful!', 'awesome!!', 'underrated', 'superbly',
'beautiful!', 'amazing!', 'excellent.', 'outstanding', 'touching',
'awesome', 'penny!', 'excellent', 'delightful', 'amazing!!', 'winner!',
'nice!', 'superb!', 'sooner!', 'childhood!', 'print...', 'favorite!',
'haunting', 'thoughtful', 'great!!!', 'riveting', 'fantastic',
'provoking', 'excelente', 'underappreciated', 'wonderfully',
'invaluable', 'liger', 'excellent!!!', 'flawless', 'must-have',
'chilling', 'finally!', 'gem!', 'kicks', 'indispensable', 'exceeded',
'addictive!', 'another!', 'hilarious!', 'enchanting', 'delicious!',
'excellent!!', 'best!!', 'cute!', 'brilliant', 'superb', 'rocks!!!',
'hit!', 'thought-provoking', 'wonderful!!', 'classic!', 'yummy',
'awsome!', 'sublime', 'wonderful', 'must!', 'refreshing', 'yummy!',
'awesome!!!', 'pleasantly', 'have!', 'life-changing', 'solid,',
'wonderful!!!', 'best!!!', 'yay', 'terrific!', 'awesome!!!!',
'wonderfull', 'quintessential', 'fabulous.', 'unbeatable',
'eye-opening', 'unforgettable', 'captivating', 'remembering',
'spellbinding', 'awsome', 'finally,', 'heaven!', 'classy', 'thriller!',
'fascinating!']

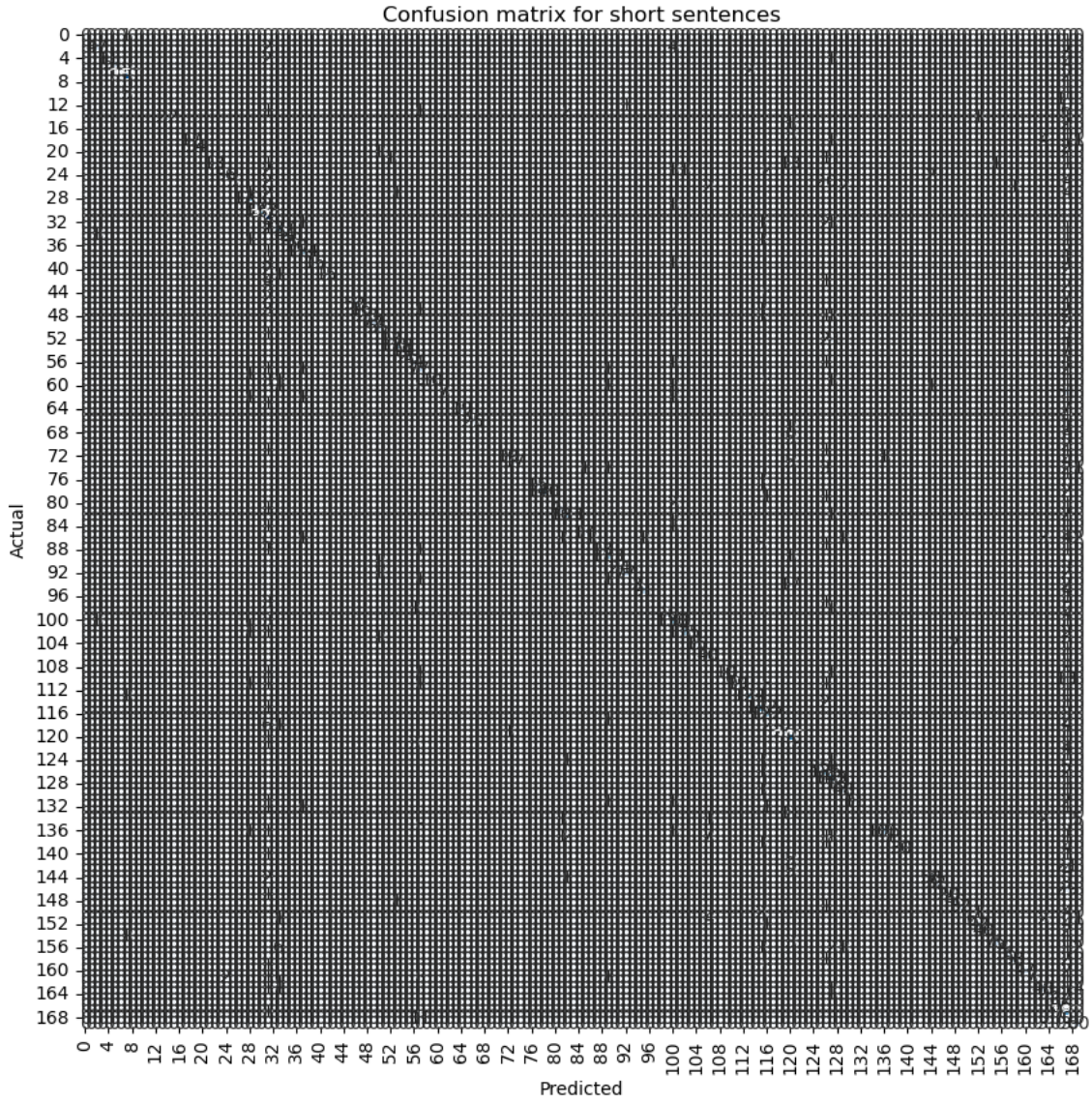
14. Setting ‘min_df’ to 2 can be reasonable especially when you want to filter out very rare terms.

15. accuracy: 0.804

16.
average sentence length in characters: 126.436
accuracy for short sentences: 0.851
accuracy for long sentences: 0.936

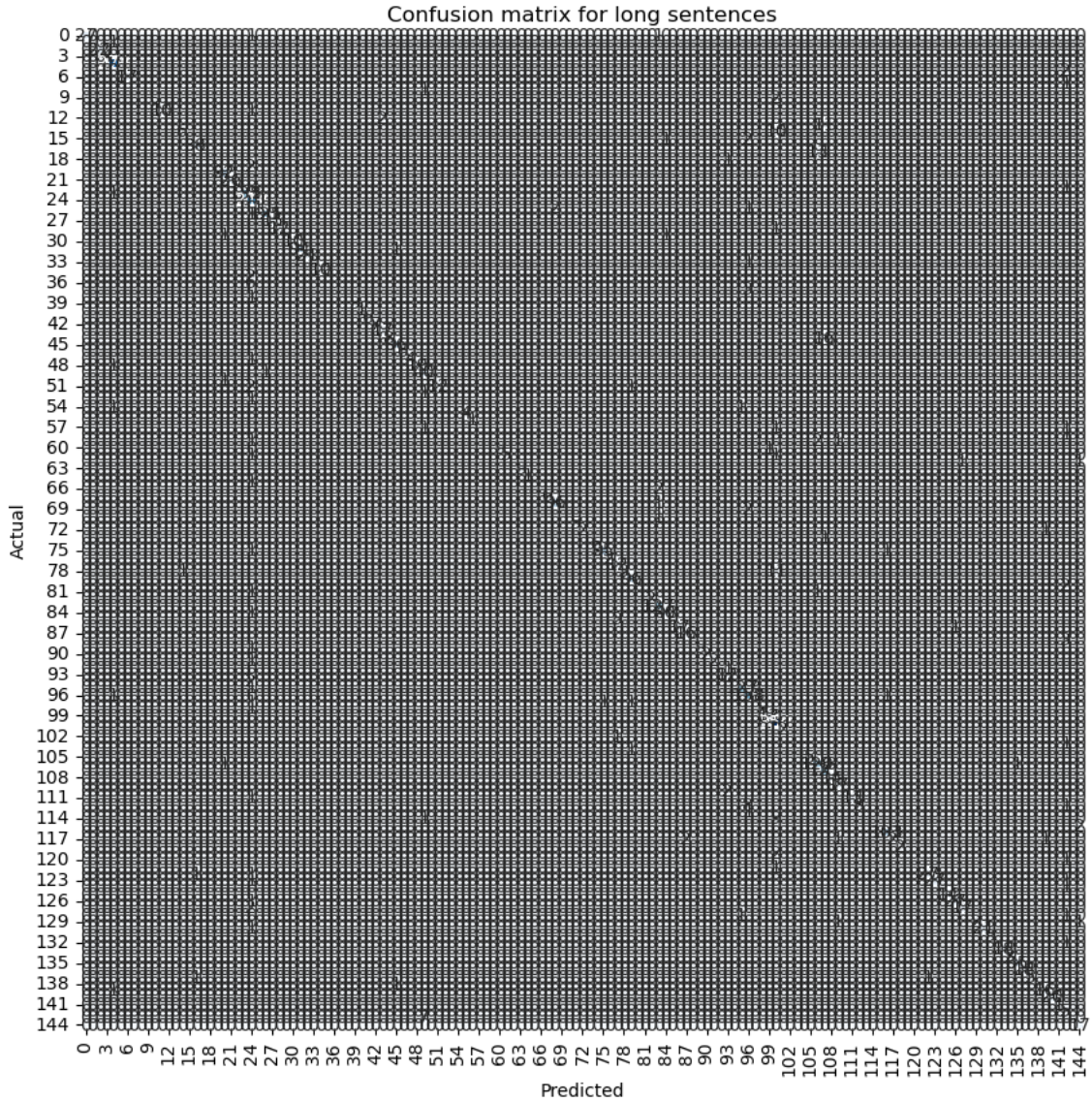
The accuracy is not the same for two groupes, it is higher for long sentences.

17.



classificaiton report for short sentences:

	precision	recall	f1-score	support
ace_Arab	0.00	0.00	0.00	1
aeb_Arab	0.00	0.00	0.00	1
afr_Latn	0.96	0.87	0.91	54
ajp_Arab	0.00	0.00	0.00	3
als_Latn	1.00	0.79	0.88	14
amh_Ethi	1.00	0.93	0.96	55
apc_Arab	0.00	0.00	0.00	5
arb_Arab	0.96	0.99	0.98	258
ars_Arab	0.00	0.00	0.00	1
ary_Arab	0.00	0.00	0.00	1
arz_Arab	0.00	0.00	0.00	6
asm_Beng	0.00	0.00	0.00	11
awa_Deva	0.00	0.00	0.00	1
ayr_Latn	0.00	0.00	0.00	10
azj_Latn	1.00	0.92	0.96	24
bak_Cyrl	0.00	0.00	0.00	3
bam_Latn	0.00	0.00	0.00	1
bel_Cyrl	0.00	0.00	0.00	3
bem_Latn	1.00	0.71	0.83	24
ben_Beng	1.00	0.94	0.97	36
bho_Deva	0.00	0.00	0.00	3
bos_Latn	0.00	0.00	0.00	13
...				
accuracy			0.85	6026
macro avg	0.53	0.45	0.47	6026
weighted avg	0.85	0.85	0.83	6026



```
classificaiton report for long sentences:
              precision    recall  f1-score   support

    afr_Latn       1.00      0.93      0.96        29
    ajp_Arab       0.00      0.00      0.00         1
    als_Latn       1.00      1.00      1.00        22
    amh_Ethi       1.00      1.00      1.00         4
    arb_Arab       0.98      1.00      0.99       274
    asm_Beng       0.00      0.00      0.00         2
    azj_Latn       1.00      1.00      1.00        17
    bak_Cyrl       0.00      0.00      0.00         1
    ban_Latn       0.00      0.00      0.00         1
    bel_Cyrl       0.00      0.00      0.00         2
    bem_Latn       1.00      1.00      1.00         3
    ben_Beng       1.00      0.91      0.95        11
    bho_Deva       0.00      0.00      0.00         2
    bos_Latn       0.00      0.00      0.00        11
    bul_Cyrl       0.86      0.38      0.52        16
    cat_Latn       0.00      0.00      0.00         3
    ceb_Latn       0.95      1.00      0.97        38
    ces_Latn       0.00      0.00      0.00        13
    ckb_Arab       0.00      0.00      0.00         1
    cym_Latn       0.00      0.00      0.00         2
    dan_Latn       0.97      1.00      0.98        92
    deu_Latn       1.00      1.00      1.00        26
...
    accuracy                           0.94      3565
   macro avg       0.61      0.58      0.59      3565
weighted avg       0.91      0.94      0.92      3565
```

- There are diagonal elements and off-diagonal elements on the confusion matrix. The diagonal elements are the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabeled by the classifier. The higher the diagonal values of the confusion matrix the better, indicating many correct predictions.
- The weighted average is higher than the macro average in the two cases. It suggests that the performance metrics are influenced more by the larger classes in terms of the number of instances. To improve the performance, the model's performance across all classes should be more balanced.

18. This line of code creates a set (scripts_icu) containing the names of Unicode scripts for each character in the example. The set will only contain unique script names, as sets do not allow duplicate elements.

19. accuracy: 0.793
I removed all non-latin scripts from the dataset and trained the model, but there was no significant change in accuracy.