# SI 630 Project Proposal

**Yu Yan**

## 1 Abstract

In this project, I explored two different tasks related to natural language processing, keyword extraction, and rating score prediction. For the keyword extraction task, I compared two methods, Baseline and Keybert extraction, for extracting one-word and two or one-word keywords from movie reviews. I evaluated the performance of both methods using precision, recall, and B-cubed F1 score. The results showed that the Baseline method performed better than the Keybert extraction method for both types of keywords, and one-word keywords performed better than two or one-word keywords for both methods. For the rating score prediction task, I compared three machine learning models, SVM, Naïve Bayes, and MiniLM-L12, for predicting the rating score of a movie based on its reviews. I evaluated the performance of all models using MSE, RMSE, MAE, Accuracy, and F1 score. The results showed that MiniLM-L12 performed the best among all models. Finally, I also compared the performance of MiniLM-L12 with and without using the keywords sentence, and the result showed that using MiniLM-L12 with full reviews results in significantly better performance than using MiniLM-L12 with only the keywords sentence. Overall, our results suggest that using state-of-the-art language models like MiniLM-L12 can be very effective for text-related tasks, and using just the keywords sentence to represent a full text may not be a reasonable approach in many cases.

## 2 Introduction

Movies have always been loved by people, and I am one of the many movie fans. Many times, when I browse movies on the Internet, it will introduce the genres, director, and rating of the movie. I especially like the IMDB website, on this website, we can clearly see the keywords of each movie. By browsing the keywords, I will not be seriously spoiled, but also understand most of the general content of the movie. So, I went to find out how IMDB obtained these keywords and was surprised to find that these keywords were actually made by manual typing. I decided to do something for IMDB through my recent learning of NLP. Although I can't get the movie dialogue content directly, I can see the user's comments on the movie, and the user's comments on the movie may help the movie automatically generate keywords. Therefore, this project will integrate users' comments on movies and extract keywords from them, then compare the extracted keywords with those manually entered on IMDB to verify whether we can directly use review contents to generate the keywords for movies. Besides, I also in interested in using movie review contents to predict the rating scores. Furthermore, I want to know whether I can use the keywords sentence to predict the rating scores. Therefore, there will be three parts in this project, for the first one, I will implement keyword extraction algorithm, such as BERT to retrieve the keywords in the review contents, the baseline in this part will be the most frequent words in each movie reviews, and I will use B-cubed score to evaluate the result; for the second part, I will use both classification and regression algorithms to predict the rating scores by the review contents, the baseline for this part will be the average rating scores for all reviews in the training dataset, and I will use MSE, RMSE, MAE and accuracy to measure the result; for the last part, I will use MSE, RMSE, MAE to measure the result, and compare the results with I got in the second part.

The difference between what I did and others is that, no one has implemented keywords extraction on movie reviews, and they also never try to test whether we can use the keywords to replace the IMDB keywords to improve efficiency; therefore, it is a very new topic that no one has explored before and I think IMDB website might care about the result of my project, because it might could reduce the step for typing keywords.

In conclusion, my results showed that we cannot simply rely on the information extracted by a keywords extraction algorithm to replace the manually inputted keywords in IMDB. Additionally, using state-of-the-art language models like MiniLM-L12 can be highly effective for text-related tasks, and representing a full text using only the keywords sentence may not be a reasonable approach in many cases.

## 3 Data

For the review data, I retrieve it from: (https://ieee-dataport.org/open-access/imdb-movie-reviews-dataset). For the keyword data, I retrieve it from: (http://hiertags-beta.elte.hu/static/data/keywords.list.gz).

For review data: I first use python to combine the review CSV files together, and give each of them a new column called movie, which will show the name of the movie. Next, I convert the TXT file to CSV file, and use groupby function to combine the keywords for each movie. Then, I drop the null value for each row in the dataset and change the type rating from object to int. The mean value of the movie rating is 7.19. And here is the distribution for the rating score, we can see most of rating score is 10.

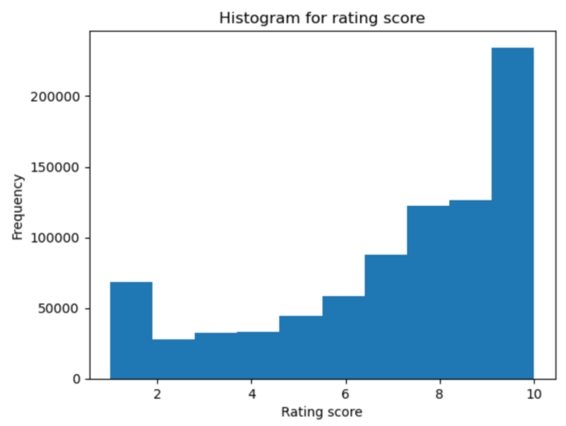


Figure 1: Histogram for rating score

For keywords data: I use csv.reader and csv.writer to rewrite the file, and make it from txt file to a csv file, which I can use to do further analysis. Next, I change the type of key to string and use strip function to delete some special characters. After that, I also use groupby function to combine the keywords for each movie.

Combine the two datasets: I merge the two datasets by the movie name using inner join, and delete the movies which has less than 100 reviews, and finally get the data I want. This dataset contains 750880 rows and 9 columns, without null value in review, movie title and keyword, I think it is fine for further analysis.

## 4 Related Work

### 4.1 Search queries related to COVID-19 based on keyword extraction

Paper (Kelebercová and Munk, 2022) presented how to use the keyword extraction model to obtain the most trending searching queries related to COVID-19 and then use the keywords to determine whether the information is true or false. Due to COVID-19, it has become increasingly difficult to distinguish between true and false information on social media, which has seriously affected people's quality of life. Therefore, the author uses 3119 articles that have been marked as true and false news as data sources to analyze fake news under COVID-19. They use the NLTK library to organize the data and convert words into tokens. Next, the author performed keyword extraction through the KeyBERT package and only extracted nouns to make keywords. In order to further prove the accuracy of these keywords, the authors extracted information about COVID-19 keywords from Google Trends, compared the keywords they extracted with the keywords generated in Google Trends, and eliminated a small number of irrelevant keywords. They first adopt non-parametric Spearman Rank Correlation to measure the statistical dependence between binary false news variable and aggregated (mean, median) top query and rising query values from queries that relate to the extracted keywords. Next, the Mann-Whitney U Test and robust descriptive statistics were also used to explore the relationship between the two. The results of the research showed that the strategies they adopted can be used to identify fake news in the media.

### 4.2 Automatic Keyword Extraction on Twitter

Paper (Marujo et al., 2015) built a corpus of tweets annotated with keywords, which was used to build and evaluate a system to automatically extract keywords on Twitter. The researchers experimented with various methods, including TFIDF, MAUI(Default), MAUI(WordVectors), MAUI(Brown), and MAUI(Brown+WordVectors), and found that MAUI(Brown+WordVectors) was the most effective. In terms of evaluation, they utilized the F1 score, which measures precision as the proportion of extracted keywords that are accurate, and recall as the proportion of correctly extracted keywords compared to the total number of keywords in the dataset.

### 4.3 Keyword extraction, ranking, and organization for the neuroinformatics platform

Paper (Usui et al., 2007) aimed to develop a tool that automates the entire process of term extraction and keyword/topic identification because the process of identifying these appropriate keywords relies on the availability of human experts, which would spend a lot of resources. The author used the collections of research abstracts from 1992 to 2004 of the Vision Research (VR) and Investigative Ophthalmology and Visual Science (IOVS) journals as test cases. They first implemented the TF–IDF method to get the keywords and found this method would cause some errors, for example, keywords often appear in documents that deal with similar topics. Therefore, the author introduced another method called TF–ITF–TDCF to retrieve the keywords. After the keywords extraction, the author used precision and recall to evaluate their models, presenting that the TF–ITF–TDCF model achieved the best performance.

### 4.4 Keyword Extraction for Social Snippets

Paper (Li et al., 2010) conducted experiments using different classification models to extract keywords from social media texts. The training and testing data used in the experiments consisted of 1830 Facebook status updates. The researchers used GBM, DT, SVM, LR and TFIDF models to do the keywords extraction part, and they found GBM performed best. Besides, they also found that use top-p

### 4.5 Performance Evaluation of Keyword Extraction Methods and Visualization for Student Online Comments

Paper (Liu et al., 2020) presented several methods to extract the keywords for the online comments of students, which would be beneficial to lecturers. The author collected student comment data over a period of two years and used the dataset to extract the topic keywords from the text based on the existing algorithms. They implemented NB, LogR, SVM, CNN, and Att-LSTM algorithms for this research, and used F1 score and accuracy to evaluate the results. The results showed that deep learning algorithms achieved great accuracy, but with more training time, and Att-LSTM performs the best in terms of all the metrics used.

### 4.6 Prediction of rating based on review text of Yelp reviews

Paper (Channapragada and Shivaswamy, 2015) demonstrates predicting ratings through Yelp reviews. The authors use both classification and regression models for prediction, with the Naive Bayes classification model showing best results. Additionally, the authors aimed to perform sentiment classification for the reviews, categorizing them into funny, useful, or cool. However, evidence suggests that it is not possible to determine emotional nuances in review text alone.

## 5 Method

### 5.1 Keywords extraction part

For this part, I first create a dataframe called baseline, which uses groupby function to group all of the reviews together with the same movie title and keywords. Next, I drop the duplicates elements in key column for each row, delete null and nan value, and replace the “-” with “ ”.

Due to the variable length of keywords, extracting keywords that are too long may lead to disappointing results. Therefore, I limited the length of keywords to 1-2 words. Next, since each keyword is separated by a comma, I counted the number of commas in each row of keywords, added 1, and placed the resulting number as the number of movie keywords in the dataframe. I named this column "num_key". Then, I removed rows with a count of 1 (i.e., movies with no keywords consisting of 1-2 words). Finally, I established stopwords to avoid extracting meaningless keywords during keyword extraction.

After that, I first defined the function "get_top_words_2" to extract the most frequently occurring words as keywords. I created a new string from the "text" variable by converting all characters to lowercase and only including alphanumeric and whitespace characters. Next, I removed stopwords and counted the remaining words. Then, I combined adjacent words into pairs and counted the resulting new 2-word keywords. Finally, I selected the top "num_key" keywords with the highest frequency from the two sets of results and used them as the keywords for this review.

```
100%|██████████| 889/889 [10:14<00:00,  1.45it/s]
```

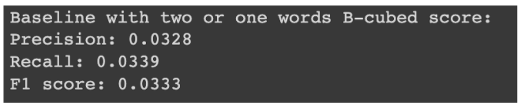
Figure 2: The process of obtaining the baseline

In addition, I also used KeyBERT (https://github.com/MaartenGr/KeyBERT) to extract keywords. I chose the "valurank/MiniLM-L6-Keyword-Extraction" (https://huggingface.co/valurank/MiniLM-L6-Keyword-Extraction) model, set the "keyphrase_ngram_range" to 1-2, and ignored stopwords. I selected the top "num_key" keywords closest to the text as the results, and stored them in a column named "KeyBERT_keywords".

```
100%|██████████| 889/889 [3:51:54<00:00, 15.65s/it]
```

Figure 3: The process of Keybert keyword extraction

Next, to calculate the B-cubed score, I converted each row in the "key" column to a list and defined a function called "calculate_b_cubed_precision_recall_f1_score" to calculate the precision, recall, and B-cubed F1 score. Firstly, I calculated the baseline B-cubed F1 score by converting the "keywords" and "predicted keywords" to sets and passing them to the function I defined. In the function, I used the "intersection" function to calculate the number of correctly predicted keywords divided by the total number of predicted keywords as precision. Then, I calculated the number of correctly predicted keywords divided by the total number of keywords given by IMDB as recall. Finally, I calculated the F1 score of that row using a formula. Next, I calculated the average precision, recall, and B-cubed F1 score for all movies, and here are the results.

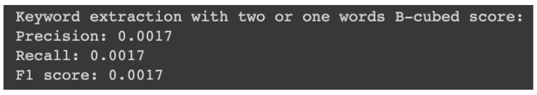


```
Baseline with two or one words B-cubed score:
Precision: 0.0328
Recall: 0.0339
F1 score: 0.0333
```

Figure 4: B-cubed score for two or one-word baseline

For using keybert model part, I get the result below.

It seems that baseline performs better than keybert model. After check the result of keybert model, I find most of the keywords containing two letters,

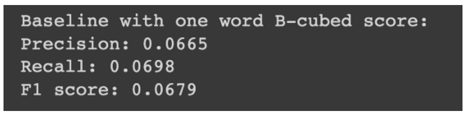


```
Keyword extraction with two or one words B-cubed score:
Precision: 0.0017
Recall: 0.0017
F1 score: 0.0017
```

Figure 5: B-cubed score for two or one-word KeyBert

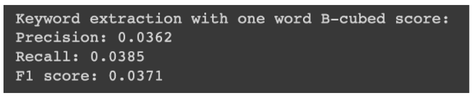
and this is reasonable, because keybert model extract the text which has the most similar content (vector) compared with the original text, and two words are easier to represent whole text. Therefore, I guess if I just choose one word by keybert model, and compare with the IMDB one-word keywords, the result might be better than now.

After processing the dataset, I obtained a new dataframe. The column key_1 represents the one-word keywords in the IMDB dataset, and the column num_key_1 represents the number of one-word keywords for each movie. The column top_words_1 is used as a baseline, which consists of the most frequent one-word keywords for each review, with a total of num_key_1 word. Finally, the column KeyBERT_keywords_1 represents the one-word keywords extracted by KeyBERT. The results are shown below:



```
Baseline with one word B-cubed score:
Precision: 0.0665
Recall: 0.0698
F1 score: 0.0679
```

Figure 6: B-cubed score for one-word baseline



```
Keyword extraction with one word B-cubed score:
Precision: 0.0362
Recall: 0.0385
F1 score: 0.0371
```

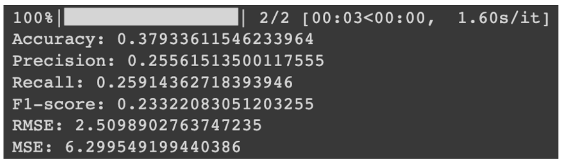
Figure 7: B-cubed score for one-word KeyBert

### 5.2 Rating score prediction part

For this part, I want to use both the classification model and regression model to predict the results, we can easily find that

I first implement Naive Bayes model to do classification for this part, I first split the dataset into two parts, one for training and the other for testing. I use sklearn library for this model, I first use CountVectorizer and fit_transform to vectorize the data. Next, I fit the naive bayes model in 2 epochs. The result is shown below.

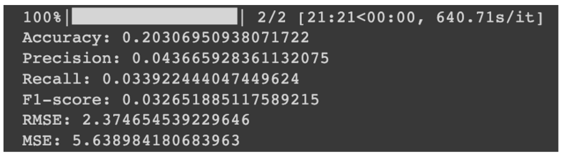
Next, I try SVM for this part, after two epochs, the result shows that the accuracy and F1 score are lower than Naive Bayes model, however the RMSE and MSE are also lower than Naive Bayes



```
100%|████████████████| 2/2 [00:03<00:00,  1.60s/it]
Accuracy: 0.37933611546233964
Precision: 0.25561513500117555
Recall: 0.25914362718393946
F1-score: 0.23322083051203255
RMSE: 2.5098902763747235
MSE: 6.299549199440386
```

Figure 8: Naive Bayes model result

model. This result shows that Naive Bayes model is better than SVM for this part, because F1 score and accuracy is often used for classification part, not RMSE and MSE.



```
100%|████████████████| 2/2 [21:21<00:00, 640.71s/it]
Accuracy: 0.20306950938071722
Precision: 0.043665928361132075
Recall: 0.033922444047449624
F1-score: 0.032651885117589215
RMSE: 2.374654539229646
MSE: 5.638984180683963
```

Figure 9: SVM model result

Then, I test the baseline for this part, I use the train dataset average rating score as the baseline, which is 7.178913, and I also plot the training dataset rating score as histogram, I show it below.

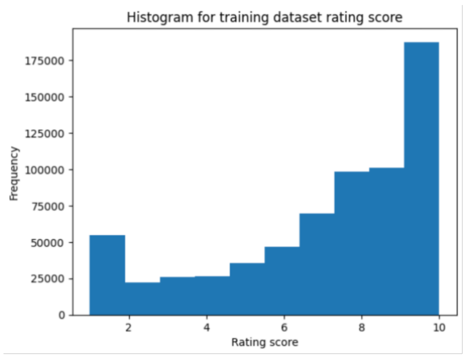
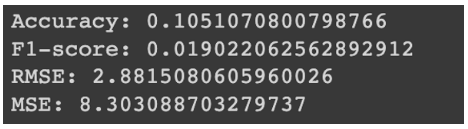


Figure 10: Histogram for training dataset rating score

Here is the result I get using baseline. I use round to get the accuracy and F1 score. And the result shows that the accuracy and F1 score are lower than I got by the two classification models and the RMSE and MSE are higher than them, but it seems that the RMSE is closed to what I got before. Anyway, we can find that both the models performed better than the baseline.

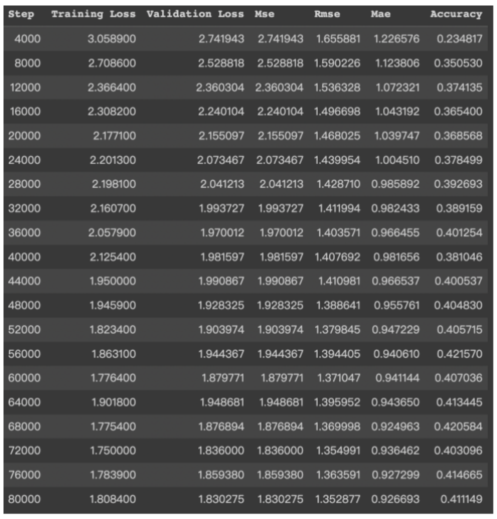
Finally, I implement a regression model to predict the rating score. I use a pre-trained model for this regression part, which is called "microsoft/MiniLM-L12-H384-uncased"(`https://huggingface.co/microsoft/MiniLM-L12-H384-uncased`). For this part, I first split the dataset to two parts, one for training and the other for testing, the tokenizer



```
Accuracy: 0.1051070800798766
F1-score: 0.019022062562892912
RMSE: 2.8815080605960026
MSE: 8.303088703279737
```

Figure 11: Baseline result

using the AutoTokenizer and AutoModelForSequenceClassification classes. Next, I define a function called preprocess_function to tokenizer the text and change the label to float type for both training and testing datasets. Furthermore, I define a function called compute_metrics_for_regression to calculate the MSE, RMSE, MAE and accuracy (by round the result) for the test dataset. And I set learning rate to 1e-5 with 16 batch sizes and training 2 epochs. The result I will show below.



| Step | Training Loss | Validation Loss | Mse | Rmse | Mae | Accuracy |
|---|---|---|---|---|---|---|
| 4000 | 3.058900 | 2.741943 | 2.741943 | 1.655881 | 1.226576 | 0.234817 |
| 8000 | 2.708600 | 2.528818 | 2.528818 | 1.590226 | 1.123806 | 0.350530 |
| 12000 | 2.366400 | 2.360304 | 2.360304 | 1.536328 | 1.072321 | 0.374135 |
| 16000 | 2.308200 | 2.240104 | 2.240104 | 1.496698 | 1.043192 | 0.365400 |
| 20000 | 2.177100 | 2.155097 | 2.155097 | 1.468025 | 1.039747 | 0.368568 |
| 24000 | 2.201300 | 2.073467 | 2.073467 | 1.439954 | 1.004510 | 0.378499 |
| 28000 | 2.198100 | 2.041213 | 2.041213 | 1.428710 | 0.985892 | 0.392693 |
| 32000 | 2.160700 | 1.993727 | 1.993727 | 1.411994 | 0.982433 | 0.389159 |
| 36000 | 2.057900 | 1.970012 | 1.970012 | 1.403571 | 0.966455 | 0.401254 |
| 40000 | 2.125400 | 1.981597 | 1.981597 | 1.407692 | 0.981656 | 0.381046 |
| 44000 | 1.950000 | 1.990867 | 1.990867 | 1.410981 | 0.966537 | 0.400537 |
| 48000 | 1.945900 | 1.928325 | 1.928325 | 1.388641 | 0.955761 | 0.404830 |
| 52000 | 1.823400 | 1.903974 | 1.903974 | 1.379845 | 0.947229 | 0.405715 |
| 56000 | 1.863100 | 1.944367 | 1.944367 | 1.394405 | 0.940610 | 0.421570 |
| 60000 | 1.776400 | 1.879771 | 1.879771 | 1.371047 | 0.941144 | 0.407036 |
| 64000 | 1.901800 | 1.948681 | 1.948681 | 1.395952 | 0.943650 | 0.413445 |
| 68000 | 1.775400 | 1.876894 | 1.876894 | 1.369998 | 0.924963 | 0.420584 |
| 72000 | 1.750000 | 1.836000 | 1.836000 | 1.354991 | 0.936462 | 0.403096 |
| 76000 | 1.783900 | 1.859380 | 1.859380 | 1.363591 | 0.927299 | 0.414665 |
| 80000 | 1.808400 | 1.830275 | 1.830275 | 1.352877 | 0.926693 | 0.411149 |

Figure 12: Training process for MiniLM-L12 model

### 5.3 Rating score prediction by keywords sentence part

For this part, I want to check whether the keyword sentence can be used to predict the rating score. Therefore, I first choose part of reviews, which has length between 5000 and 5400, that is because I just want to use 10 to 40 keywords to represent this sentence, and I think 5000 to 6000 is a suitable length to conclude by 10 to 40 keywords; and another reason is that the keywords extraction step works very slowly, use all of the reviews will cost about 1000 hours. After that, I get 6437 reviews to do the training and testing part. Here is the histogram for the part of review rating score. The distribution is similar to the total reviews.

I first use KeyBERT and choose "valurank/MiniLM-L6-Keyword-Extraction"

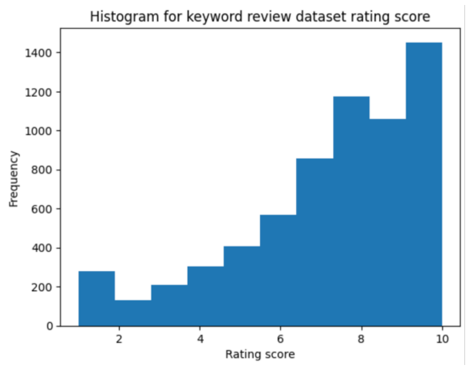
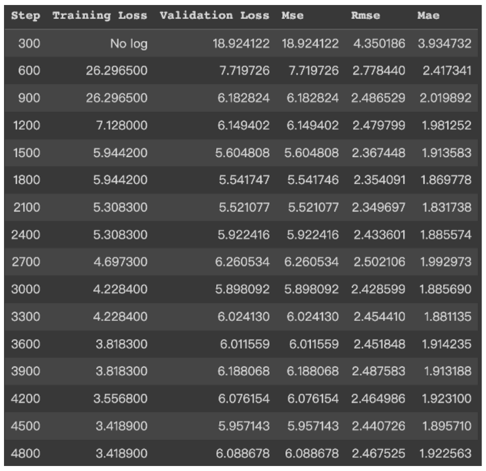


Figure 13: Histogram for keyword review dataset rating score



| Step | Training Loss | Validation Loss | Mse | Rmse | Mae |
|---|---|---|---|---|---|
| 300 | No log | 18.924122 | 18.924122 | 4.350186 | 3.934732 |
| 600 | 26.296500 | 7.719726 | 7.719726 | 2.778440 | 2.417341 |
| 900 | 26.296500 | 6.182824 | 6.182824 | 2.486529 | 2.019892 |
| 1200 | 7.128000 | 6.149402 | 6.149402 | 2.479799 | 1.981252 |
| 1500 | 5.944200 | 5.604808 | 5.604808 | 2.367448 | 1.913583 |
| 1800 | 5.944200 | 5.541747 | 5.541746 | 2.354091 | 1.869778 |
| 2100 | 5.308300 | 5.521077 | 5.521077 | 2.349697 | 1.831738 |
| 2400 | 5.308300 | 5.922416 | 5.922416 | 2.433601 | 1.885574 |
| 2700 | 4.697300 | 6.260534 | 6.260534 | 2.502106 | 1.992973 |
| 3000 | 4.228400 | 5.898092 | 5.898092 | 2.428599 | 1.885690 |
| 3300 | 4.228400 | 6.024130 | 6.024130 | 2.454410 | 1.881135 |
| 3600 | 3.818300 | 6.011559 | 6.011559 | 2.451848 | 1.914235 |
| 3900 | 3.818300 | 6.188068 | 6.188068 | 2.487583 | 1.913188 |
| 4200 | 3.556800 | 6.076154 | 6.076154 | 2.464986 | 1.923100 |
| 4500 | 3.418900 | 5.957143 | 5.957143 | 2.440726 | 1.895710 |
| 4800 | 3.418900 | 6.088678 | 6.088678 | 2.467525 | 1.922563 |

Figure 14: Histogram for keyword review dataset rating score

as model to extract the keywords for each review. Besides, I also set keyphrase_ngram_range between 10 and 40 to make sure that the keywords sentence will be between 10 and 40, and the top_n is set to 1, which means I will just use one most relevant keywords sentence to represent the total review.

Next, I implement a regression model to predict the rating score. I use a pretrained model for this regression part, which is called "microsoft/MiniLM-L12-H384-uncased". For this part, I first split the dataset to two parts, one for training and the other for testing, the tokenizer using the AutoTokenizer and AutoModelForSequenceClassification classes. Next, I define a function called preprocess_function to tokenizer the text and change the label to float type for both training and testing datasets. Furthermore, I define a function called compute_metrics_for_regression to calculate the MSE, RMSE and MAE for the test dataset. And I set learning rate to 1e-5 with 16 batch sizes and training 15 epochs. Finally, I get the result.

## 6 Evaluation

### 6.1 Keywords extraction part

The table provides the evaluation metrics for two different methods, the Baseline and Keybert extraction, for two different types of keywords, two or one words, and one word. The evaluation metrics used are Precision, Recall, and B-cubed F1 score. It can be observed that the Baseline method performs better than the Keybert extraction method for both types of keywords. The precision, recall, and F1 score for the Baseline method are much higher than that of the Keybert extraction method. Additionally, it can also be observed that the performance of both methods is better for one-word keywords

|  | Precision | Recall | B-cubed F1 score |
|---|---|---|---|
| Baseline for two or one words | 0.0328 | 0.0339 | 0.0333 |
| Keybert extraction for two or one words | 0.0017 | 0.0017 | 0.0017 |
| Baseline for one word | 0.0665 | 0.0698 | 0.0679 |
| Keybert extraction for one word | 0.0362 | 0.0385 | 0.0371 |

Figure 15: Evaluation for Keywords extraction part

than two or one-word keywords. The Baseline method has a higher precision, recall, and F1 score for one-word keywords than two or one-word keywords. The Keybert extraction method also shows a similar trend, but the difference is not as significant. In summary, the Baseline method performs better than the Keybert extraction method for both types of keywords, and one-word keywords perform better than two or one-word keywords for both methods, and all of the results are not great. Therefore, we cannot just use the review keywords to replace the actual keywords in IMDB.

### 6.2 Rating score prediction part

To make the result looks clearly, I make a table shows below.

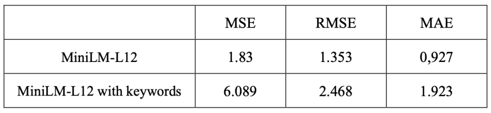
|  | MSE | RMSE | MAE | Accuracy | F1 score |
|---|---|---|---|---|---|
| Baseline | 8.303 | 2.882 |  | 0.105 | 0.019 |
| SVM | 5.639 | 2.375 |  | 0.203 | 0.033 |
| Naïve Bayes | 6.3 | 2.51 |  | 0.379 | 0.233 |
| MiniLM-L12 | 1.83 | 1.353 | 0,927 | 0.411 |  |

Figure 16: Evaluation for Rating score prediction part

Based on the table, we can find that the pretrained model MiniLM-L12 performs better than baseline, SVM and Naïve Bayes, which has lowest score in MSE and RMSE, and highest score on accuracy. Besides, the SVM and Naïve Bayes models also perform better than the baseline, which contains the worst score in this comparison.

### 6.3 Rating score prediction by keywords sentence part

I just want to compare the model works with full reviews and just the keywords sentence. Therefore, I make a table below.



| | MSE | RMSE | MAE |
|---|---|---|---|
| MiniLM-L12 | 1.83 | 1.353 | 0,927 |
| MiniLM-L12 with keywords | 6.089 | 2.468 | 1.923 |

Figure 17: Evaluation for Rating score prediction by keywords sentence part

The table shows that using MiniLM-L12 with full reviews results in significantly better performance than using MiniLM-L12 with only the keywords sentence, because MSE, RMSE and MAE are lower than using only the keywords sentence. The reason I want to compare the keywords sentence with the full text is I want to know whether just use keywords to summary full reviews can get the same or just a little lower result, therefore, KeyBert might can be used to summary a text without losing too much information (at least in this situation). However, the result shows that it is not reasonable.

## 7 Discussion

For part 1, as the B-cubed F1 scores show, the result is not fine; therefore, we cannot directly retrieve the keywords from reviews to represent the keywords typed in IMDB. It is a shame that we cannot help the IMDB work more efficiently. The reason I think is the most of the typed IMDB keywords is limited to a scope, and the some of the keywords is less likely to appear in reviews very frequently. Furthermore, the result shows that the Baseline method performs better than the Keybert extraction method for both types of keywords, and one-word keywords perform better than two or one-word keywords for both methods. It is clear to know that one-word keywords would perform better than two or one-word keywords, the reason is that one-word keywords are generally more frequent and occur more frequently in the text, and they are easier to extract and have a higher chance of being relevant to the text, so it is very possible to get the right keywords in reviews compared to two-words or one-word keywords. Besides, it is very confused to see that Keybert model get worse results compared with the baseline. The reason I think might be I put all of the reviews in a movie together, so the sentence will be a bit of strange which would influence the results for Keybert model, but not influence the baseline result. However, if I do not put all of the reviews in a movie together, it will spend more than 100 hours to get all of the keywords, I think I do not have enough time to do this. Anyway, it is a shame to see the result in this part.

For part2, all of the three models performed better than the baseline. The reason Naïve Bayes model performs better than SVM is because Naïve Bayes model is more suitable for text data compared with SVM model, that is because SVM needs more carefully selected features and hyperparameters, and the text is not very suitable. Besides, the SVM and Naïve Bayes models are more traditional machine learning algorithms that require feature engineering and hyperparameter tuning, they may not be as effective as language models like MiniLM-L12 that are specifically designed for natural language processing. The MiniLM-L12 model has already been trained on a large corpus of text data and is able to capture complex relationships between words and phrases, so it gets the best result. Anyway, I think the result is acceptable, and we can use the model I fine-tuned to predict the review scores.

For part 3, although the result is not very well, I think there are still some areas worth exploring, such as if better summarization of keywords can be achieved, not limited to 10-40 words, but with more accurate measures, whether the results will be improved. In addition, the small dataset may also be the reason for these issues, considering that full review training used the entire dataset while keywords sentence training only used 10% of the dataset. Besides, I think I could try just use the title of the reviews to predict the rating scores, and then compared with the result using the total review text and keyword summary text, or whether we can use the reviews to generate a title for these reviews, both of them are fine topics, I would like to try them in the future.

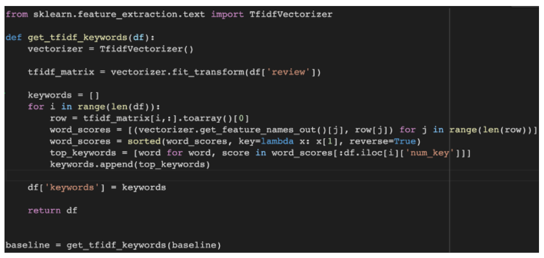
## 8 Conclusion

As a movie enthusiast, I was dissatisfied with the use of typed keywords on IMDB and therefore designed a project to improve it. Firstly, I used

Keybert to extract keywords from IMDB reviews, but compared to the baseline, I found that we cannot simply use the review keywords to replace the manually inputted keywords on IMDB. One reason is that typed keywords have certain limitations in scope, while it is difficult for review keywords to include all the restricted words. Then, I used movie reviews to predict rating scores and found that traditional machine learning models such as NB and SVM had lower performance compared to the pretrained language model microsoft/MiniLM-L12-H384-uncased, which showed good prediction results. Finally, I explored the use of keyword sentences in movie reviews to predict rating scores. Although the results were not as good as using review text, it gave me many interesting ideas, such as using review titles to predict rating scores or using review text to generate titles to construct a supervised learning model. In any case, this project was a valuable attempt to apply NLP models to previously unexplored areas. Although not all results were satisfactory, it was still a valuable attempt.

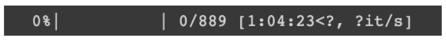
## 9 Other Things I Tried

In part 1, during the keyword extraction process, I attempted to use TF-IDF, but the code ran very slowly and there was no output for a long time. Therefore, I gave up on using the TF-IDF model to extract keywords.



```
from sklearn.feature_extraction.text import TfidfVectorizer

def get_tfidf_keywords(df):
    vectorizer = TfidfVectorizer()

    tfidf_matrix = vectorizer.fit_transform(df['review'])

    keywords = []
    for i in range(len(df)):
        row = tfidf_matrix[i,:].toarray()[0]
        word_scores = [(vectorizer.get_feature_names_out()[j], row[j]) for j in range(len(row))]
        word_scores = sorted(word_scores, key=lambda x: x[1], reverse=True)
        top_keywords = [word for word, score in word_scores[:df.iloc[i]['num_key']]]
        keywords.append(top_keywords)

    df['keywords'] = keywords

    return df


baseline = get_tfidf_keywords(baseline)
```

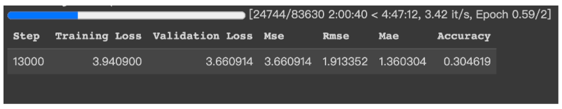
Figure 18: TF-IDF model to extract keywords



```
  0%|          | 0/889 [1:04:23<?, ?it/s]
```

Figure 19: The process of TF-IDF model for keywords extraction

Furthermore, while writing the report, I had an idea to try to use the title to predict the rating score. However, due to limited GPU resources, I had already used up all the paid resources on Colab and did not plan to continue paying, so I did not produce any specific results. Nevertheless, from the results below, we can see that even with only 0.59 epochs, the prediction results of the rating score based on the title were better than those based on the extraction of keywords and sentences. Therefore, if there were more resources available, I am curious about how close the final results would be to the parameters given in the reviews. This is a big regret for this project. I will leave this code in the last section of my ipynb file, and when I have time and GPU resources, I will continue to try and improve my project.



```
[24744/83630 2:00:40 < 4:47:12, 3.42 it/s, Epoch 0.59/2]
Step   Training Loss   Validation Loss   Mse        Rmse       Mae        Accuracy
13000  3.940900        3.660914          3.660914   1.913352   1.360304   0.304619
```

Figure 20: The training process of using title to predict rating score

In addition, if the final results obtained from the title and review are very close, I would like to train a supervised model to identify the title of a text from a review, and use the results of my own trained model to predict the rating score. This could even be a separate project in itself. Due to limited time and resources, I will try this approach during the summer. I believe this will be a very interesting project!

## 10 What You Would Have Done Differently or Next

Actually, I feel like I've accomplished a lot with limited time and GPU resources, but there are still some regrets. For the first part of the keyword extraction, if I had more time and GPU resources, I might extract keywords from each review separately instead of putting them all together, and then combine and sort the keywords of all reviews for each movie, change the length to num_key, and calculate b-cubed. I think this way would be more reasonable because each review would be an independent part and not affected by the context before and after it, which could reduce some errors generated by using KeyBert for keyword extraction.

As for using the movie title to predict the rating score, I'm very regretful that I didn't get the results due to time and resource constraints. In the future, I will try to obtain more resources to output the results and compare them with the results from the review text. If the rating score can be accurately predicted using only the title, a lot of resources will be saved. If a model that can quickly summarize reviews and generate titles can be created, I think it will be very meaningful. These are all directions and things I want to explore in the future!

## References

Sasank Channapragada and Ruchika Shivaswamy. 2015. Prediction of rating based on review text of yelp reviews.

Lívia Kelebercová and Michal Munk. 2022. Search queries related to covid-19 based on keyword extraction. *Procedia Computer Science*, 207:2618–2627. Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 26th International Conference KES2022.

Zhenhui Li, Ding Zhou, Yun-Fang Juan, and Jiawei Han. 2010. Keyword extraction for social snippets. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 1143–1144, New York, NY, USA. Association for Computing Machinery.

Feng Liu, Xiaodi Huang, Weidong Huang, and Sophia Duan. 2020. Performance evaluation of keyword extraction methods and visualization for student online comments. *Symmetry*, 12:1923.

Luís Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W. Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on Twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 637–643, Beijing, China. Association for Computational Linguistics.

S. Usui, P. Palmes, K. Nagata, T. Taniguchi, and N. Ueda. 2007. Keyword extraction, ranking, and organization for the neuroinformatics platform. *Biosystems*, 88(3):334–342. BIOCOMP 2005: Selected papers presented at the International Conference - Diffusion Processes in Neurobiology and Subcellular Biology.