EECS4404 Fall 2024

Project

December 3, 2023

He Yang Yuan
216386286
yuanh@my.yorku.ca
Ahmet Ozer
215859028
cagri32@my.yorku.ca

Objective:

The primary goal of this project is to discern the most effective methodology for classifying user reviews by conducting a comprehensive comparison between traditional Bag of Words (BOW) and advanced Text Embedding techniques. Given the diverse lengths and structures of user reviews, which fall into Positive, Neutral, and Negative categories, the project aims to identify the method that yields optimal results through rigorous training on the designated Training Dataset. The selected method is subsequently applied to predict classifications for the entire Testing Dataset. Acknowledging the subjectivity in classifying user reviews, the project recognizes that, despite achieving high training accuracy, misclassifications may still occur, particularly when subjected to subsequent human evaluations.

Text Embedding:

Setup and Configuration Details

- The experiments were conducted leveraging a Jupyter Notebook environment, utilizing a GPU powered by an RTX 2070 Super for enhanced computational performance.
- The tests were executed using CUDA and PyTorch. The sentence embeddings were generated using the Sentence Transformer library and the computation offloaded to the CUDA-enabled GPU via the 'device' parameter set to 'cuda'.
- A partition of 80% for training and 20% for testing was employed across all tests to evaluate the classification accuracy. The evaluations were conducted on a dataset comprising 60,000 reviews.
- In order to prevent the occurrence of the 'STOP: TOTAL NO. of ITERATIONS REACHED LIMIT' warning during testing, the Max Tier parameter was set to 2000.
- The configuration of other method settings, such as **`Random state = 42, hidden_layer_sizes= (100, 50), n_estimators=100`**, etc. remained consistent with the parameters employed in the previous assignment.

Evaluated Classification Algorithms

- K-NN (1),
- K-NN (3),
- Gaussian Naive Bayes,
- Decision Tree,
- Random Forest,
- Logistic Regression,
- MLP Neural Network.

Algorithm Selection and Optimization:

Comparative Analysis of Classification Performance and Efficiency

Algorithm	Classification Accuracy (%)	Running Time (seconds)
Logistic Regression	86.23	168.92
MLP Neural Network	83.53	1173.42
Random Forest	81.45	3141.24
k-NN (k=3)	78.45	97.06
k-NN (k=1)	74.23	97.16
Decision Tree	70.62	2028.83
Gaussian Naive Bayes	61.98	3.64

From the above information, it is clear that the **Decision Tree** and **Random Forest** have extreme long running times and lower accuracy rates than other methods, so they were removed in later model tests.

Performance Evaluation of Various Language Models

All-mpnet-base-v2:

Runtime 909 seconds

Algorithm	Accuracy
Logistic Regression	86.24
MLP Neural Network:	83.65
k-NN (k=3):	78.45
k-NN (k=1):	74.23
Gaussian Naive Bayes:	61.98

All-MiniLM-L12-v2

Runtime 553 seconds

Algorithm	Accuracy
Logistic Regression:	82.04
MLP Neural Network:	77.79
k-NN (k=3):	74.52
k-NN (k=1):	70.09
Gaussian Naive Bayes:	59.93

Multi-qa-mpnet-base-dot-v1:

Runtime: 1015 seconds

Algorithm	Accuracy
Logistic Regression:	87.03
MLP Neural Network:	83.90
k-NN (k=3):	78.06
k-NN (k=1):	73.99
Gaussian Naive Bayes:	62.94

Multi-qa-distilbert-cos-v1

Runtime 801 seconds

Algorithm	Accuracy
Logistic Regression:	82.69
MLP Neural Network:	79.40
k-NN (k=3):	74.55
k-NN (k=1):	70.06
Gaussian Naive Bayes:	57.44

All-distilroberta-v1

Runtime 687 seconds

Algorithm	Accuracy
Logistic Regression:	86.13
MLP Neural Network:	83.11
k-NN (k=3):	79.22
k-NN (k=1):	75.20
Gaussian Naive Bayes:	69.02

All-MiniLM-L6-v2

Runtime 546 seconds

Algorithm	Accuracy
Logistic Regression:	82.58
MLP Neural Network:	78.40
k-NN (k=3):	75.38
k-NN (k=1):	70.90
Gaussian Naive Bayes:	62.67

Identifying Optimal Language Model for Final Prediction on Test Dataset

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-mpnet-base-v2	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2	64.46	40.04	52.25	5000	43 MB
paraphrase-multilingual-MiniLM-L12-v2	64.25	39.19	51.72	7500	420 MB
paraphrase-MiniLM-L3-v2	62.29	39.19	50.74	19000	61 MB
distiluse-base-multilingual-cased-v1	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2	60.18	27.35	43.77	4000	480 MB

Based on the above results, **Logistic Regression** is the relatively optimal algorithm, but we still need to choose a relatively best model from all the other models. Comparing the accuracy of Logistic Regression, it is not difficult to see that the accuracy of models from **all-MiniLM-L12-v2** downwards is not much different, and the runtime is basically consistent.

Therefore, we decide to choose from models including and above **all-MiniLM-L12-v2**. Ultimately, based on runtime and accuracy, we decide to use the **all-mpnet-base-v2** model for our final prediction on the Test Dataset, as it has a relatively average runtime and accuracy compared to other models.

Text Embedding Prediction for Whole Test Dataset:

```
X_train_embeddings:  [[ 0.02663761  0.03765713 -0.02672531 ...  0.01316451  0.03408683
  -0.03646544]
 [-0.01771661  0.05799315  0.01048527 ...  0.00217499  0.04842884
   0.01146969]
 [ 0.02347835  0.0026954  -0.0007662  ...  0.04005559  0.0545788
  -0.00739075]
 ...
 [-0.04720769  0.00584192 -0.02096248 ... -0.0014991   0.05491195
   0.00607969]
 [ 0.01004585 -0.02565398  0.00019075 ...  0.02423306 -0.00107937
  -0.00932442]
 [-0.07179622  0.03802516  0.01490516 ... -0.02780377  0.03210668
  -0.00457317]]
['negative' 'positive' 'positive' ... 'positive' 'positive' 'positive']
```

Using **all-mpnet-base-v2 model** for prediction, runtime: 691 seconds
This is a screenshot of some predictions saved to a CSV file.

	ID	Class
2	226336	negative
3	5905814	positive
4	5975964	positive
5	3677317	positive
6	4275042	positive
7	6204097	positive
8	4249473	negative
9	5326743	positive
10	4670297	positive
11	4199582	positive
12	2308075	positive
13	1280677	positive
14	6173385	positive
15	1952560	negative
16	2861577	positive
17	5579454	positive
18	2565920	negative
19	3483233	positive
20	846371	positive
21	1898042	positive
22	4719037	negative
23	837759	positive
24	802617	positive
25	6218576	positive
26	1576143	positive
27	344736	positive
28	2451596	positive
29	5538573	negative
30	1245931	neutral
31	13778	neutral
32	448399	positive
33	1527405	positive
34	6590304	positive

Manual verification was also conducted by several people, with 5 testers each receiving 5-10 randomly selected reviews. Although classification of reviews is subjective, but the average accuracy was 70%."

Bag-of-Words (BOW):

Setup and Configuration Details

- The experiments were conducted leveraging a Jupyter Notebook environment.
- A partition of 80% for training and 20% for testing was employed across all tests to evaluate the classification accuracy. The evaluations were conducted on a dataset comprising 60,000 reviews.
- **`Random state = 42, hidden_layer_sizes= (100, 50), n_estimators=100`**, etc. remained consistent with the parameters employed in the previous assignment.
- In the IPYNB file, there are 5 blocks of code to run. They are denoted with comments by Part 1, Part 2 etc.

Evaluated Classification Algorithms

- K-NN (1),
- K-NN (3),
- Gaussian Naive Bayes,
- Logistic Regression,
- MLP Neural Network.

Algorithm Selection and Optimization:

Comparative Analysis of Classification Performance and Efficiency

Algorithm	Classification Accuracy (%)	Running Time (seconds)
Logistic Regression	83.07	34.44
MLP Neural Network	79.39	728.31
k-NN (k=3)	65.73	3062.14
k-NN (k=1)	65.01	3229.19
Gaussian Naive Bayes	64.13	13.37

From the above information, it is clear that the **K-NN Algorithm with k=1 and k=3** have extreme long running times and lower accuracy rates than other methods, so they were removed in later model tests. **Gaussian Naive Bayes** was considered no more because of the low accuracy rate.

Based on the above results, **Logistic Regression** is the relatively optimal algorithm.

Feature Selection and Model Training Report

In the process of developing a classification model for our dataset, we explored various feature selection methods to enhance model efficiency and interpretability. Our focus was on optimizing the feature set using the **SelectKBest** method with the **chi2** scoring function.

SelectKBest with chi2:

- We employed the SelectKBest feature selection method provided by scikit-learn.
- The chi2 statistical test was chosen as the scoring function.
- The goal was to identify the top features based on their chi2 scores, which measure the independence between categorical variables.

Parameter Choices

- k=2000:
- We decided to retain the top 2000 features with the highest chi2 scores.
- This parameter choice was made to strike a balance between reducing dimensionality and preserving the most informative features for our classification task.

Model Training

- We utilized Logistic Regression as our classification algorithm.
- The model was trained on the training set using the selected 2000 features after experimenting with several feature numbers like 100-500-1000-1500-2000

Training and Evaluation

Data Splitting:

- The dataset was split into training and testing sets using the train_test_split function.

Accuracy Calculation:

- The accuracy of the Logistic Regression model was calculated on the training set.
- This provided insights into the model's performance in capturing patterns within the training data.

Results and Output

Training Set Accuracy:

- The training set accuracy was found to be 86.98%, showcasing the model's performance on the data used for training.
-

Predictions on Test Set:

- Predictions were made on the test set using the trained Logistic Regression model.
- The results were saved to a CSV file ('prediction2.csv') for further analysis.

Conclusion

In summary, the combination of SelectKBest with chi2 and retaining 2000 features provided a methodical approach to feature selection. The Logistic Regression model, trained on this optimized feature set, demonstrates promising accuracy on the training set. Further fine-tuning of parameters and exploration of alternative feature selection methods could be considered for continued model improvement.

Comparison of Bag-of-Words (BOW) and Text Embedding

In our comprehensive exploration of user review classification, we conducted an in-depth comparison between traditional Bag of Words (BOW) and advanced Text Embedding techniques. The objective was to discern the most effective methodology for classifying user reviews based on a diverse range of lengths and structures.

Bag-of-Words (BOW) Approach

Feature Selection and Model Training

- Utilized scikit-learn's **SelectKBest** method with the **chi2** scoring function for feature selection.
- Retained the top 2000 features with the highest chi2 scores.
- Employed Logistic Regression as the classification algorithm.

Results and Output

- Achieved a training set accuracy of 83.07%.
- Logistic Regression demonstrated relatively optimal performance among the evaluated algorithms.

Text Embedding Approach

Feature Selection and Model Training

- Leveraged the Sentence Transformer library and CUDA-enabled GPU for text embedding.
- Explored various language models, including all-mpnet-base-v2, multi-qa-mpnet-base-dot-v1, all-distilroberta-v1, all-MiniLM-L12-v2, multi-qa-distilbert-cos-v1, and all-MiniLM-L6-v2.
- Selected all-mpnet-base-v2 as the final model based on a balance between accuracy and runtime.

Results and Output

- Achieved a training set accuracy of 86.24% using the selected language model.
- The average accuracy from manual verification was 70%, considering the subjective nature of review classification.

Comparative Analysis

Accuracy and Efficiency

- Logistic Regression in the BOW approach demonstrated an accuracy of 86.98%, while the text embedding approach achieved 86.24% accuracy with the training set.
- BOW method: 34.44 seconds runtime vs. Text Embedding method: 691 seconds runtime.
- Text Embedding showed similar accuracy at the cost of increased computational time.

Considerations

- BOW is computationally more efficient, but text embedding captures nuanced patterns, resulting in higher accuracy.

Conclusion

In conclusion, the choice between Bag-of-Words and Text Embedding depends on the balance between computational efficiency and classification accuracy. Bag-of-Words, with its simplicity and faster runtime, provides a reasonable accuracy level. On the other hand, Text Embedding techniques, specifically leveraging advanced language models like all-mpnet-base-v2, offer improved accuracy but at the expense of increased computational demands. The final selection should align with the specific requirements of the classification task and available computational resources.

Anything special need to know to run programs

For BOW file, please run the code in order, that means from the beginning all the way to the end. For text embedding, the first part of code is for method evaluation, the second part of code is for generating prediction file.