Experimenting NLP Techniques On Semantic Analysis

CS505 Fall 2023 Final Project

Jialu Li, Can Erozer

Abstract:
Sentiment analysis using Natural Language Processing (NLP) involves evaluating and interpreting subjective information in text data to understand the sentiments, opinions, or emotions expressed. This technique is widely used for analyzing social media posts, product reviews, and other user-generated content to gauge public opinion or customer sentiment. It typically involves processing and classifying text as positive, negative, or neutral. Advanced NLP models, including machine learning and deep learning approaches, are employed to capture the nuances of human language and sentiment accurately.
Our final project proposes a comparative study of text classification methods applied to Amazon Prime Video reviews. We aimed to combine the tools that we have learned during class to understand how these techniques impact the performance of different models. We examined the effectiveness of text processing techniques such as stemming and lemmatization, oversampling with generating texts through a language model, and different transformer and neural network models such as RNN, Roberta, BERT, and GPT-2 on accuracy. When deciding which path to proceed with, we experimented with different tools on our dataset and gauged the value of metrics like perplexity and accuracy. According to the values of the metrics, we proceeded with the project with the tools that yielded better results. Restating our purpose, in other words, we didn’t strive to get the best accuracy possible by using different tools, yet our real aim was to compare the effectiveness of the tools we have learned in the class.
Some results that we found are that transformer models yield higher accuracy with preprocessed text. And text generation with transformer models for oversampling is a good but inefficient way, i.e. it helps to increase the accuracy and doesn’t harm the nature of the dataset.

Experiment Setup and Decision Flow:

Note: All the details of the explanations below can be found in the “MAIN.ipynb” file.

The first look:
In the original dataset, there are nine features, which are “Id”, “ProductId”, “UserId”, “HelpfulnessNumerator”, “HelpfulnessDenominator”, “Time”, “Summary”, “Text”, and “Score”. Among these columns, “Score” is the label which includes the classes 1.0, 2.0, 3.0, 4.0, and 5.0. Even though all of these features can be used to predict the score, the only features that were used were summary and text for the sake of the aim of the project. Since the average word length of the texts in the “Summary” column is 4.84, we decided to combine the “Summary” and “Text” columns and named the new column “Merged_Text”. We thought 4.84 is a low number such that it is difficult to make a sentiment analysis on it.

Dealing with Null Values:
There were 17470 null values in the label column, so we dropped all of them. And there was one null value for each of the “Summary” and “Text” columns. We may have used imputation techniques in NLP to replace two null values. But since this would have no impact on the performance, we just dropped them.

Problem Restatement: Compare the effectiveness of different techniques learned in the class on text classification.

During class, we have spent a large amount of time on language models. We learned that language models are used to create chat bots, predict the next word in a sentence and complete half-sentences. When further exploring the dataset, we encountered imbalanced classes. So we thought we could use language models in a way that is not mentioned in the class: using language models as a way of oversampling, in other words, creating synthetic data for minority classes, such as 1.0 and 2.0, to be able to work on a balanced dataset. LLMs like GPT-2 are known for their ability to generate texts near spoken language. And we thought it would be interesting to observe the ability of GPT-2 to generate texts for bad reviews of movies.

Because the reviews are user-written in an informal manner, text preprocessing is crucial before beginning to generate texts. When working on text preprocessing, we thought the depth of the level of preprocessing shouldn’t be the same for text classification and text generation since they are two different tasks. So we made two different versions of text processing with different levels of detail for the sake of the nature of the task.

Preprocessing for Text Generation:

The use of words in the text is important to reflect the sentiment. But the use of punctuation marks and capital letters is also important to capture the emotion of the text. So we just removed the punctuation marks that don’t hold any emotion, like “#$%&()*+<=>@^_[]{|}~]+”. This was essential in terms of getting rid of weird use of punctuation marks as well, like “"type-casting"”. Also, we removed numbers from the text because numbers were also used in a weird way. We removed all possible HTML tags included in the text. We didn’t do any stemming or lemmatization because LLMs can be fine-tuned with texts without applying them. (see “text_processing_lm()” function in the “MAIN.ipynb” file.)

Oversampling:

We mentioned that there was a great imbalance between classes. We want to generate texts for the minority classes, such as 1.0 and 2.0, but there should be a limit to the number of generated texts. For example, we can’t make the dataset balanced with generating texts in a way that all the classes have the same number of the majority class 5.0, i.e. we can’t generate 50k+ samples. So considering the training time, we decided to make the dataset balanced by using 10k samples from each class. In order to do this, we approximately generated samples in sizes of ⅓ of their original sample size, i.e., oversampled classes 1.0 and 2.0 so that they both have 10k samples at the end.

```
5.0    65313
4.0    27817
3.0    14482
1.0     7360
2.0     7309
```

We decided to use the GPT-2 model since it is more apt to generate texts than other transformer models like BERT due to its decoder-only architecture.
We first extracted samples that belong to class 1.0 and then fine-tuned the pre-trained GPT-2 model with these samples to generate texts. The details can be found in the “Final_project_GPT2_for_oversample_class1.ipynb” file.

We did the same thing for class 2.0. The details can be found in the “Final_project_GPT2_for_oversample_class2.ipynb” file.

For class 1.0, here are the original reviews,

```
Here are example of ten sentences from the original text:
0: Quite possibly the worst movie I have ever seen. I cannot believe this movie got made, with decent actors no less. Don't waste your time watching this
1: Could not get interested in this replacement for two really good series I managed to watch a little over half of the series pilot during it's first a
2: I hate to give this BILGE  Star! I saw this one on television recently, and if there was ever a no star rating this would get it At least from me Per
3: Just say NO! I could not agree more with the others here. Paramount is trying to get rich quick off of this series. I had every intention of buying a
4: BO.RING! I wish I could say this movie was like watching paint dry but that would've been more exciting. It's just about an alcoholic private investi
5: Yawn! Identity Thief is a Time Thief I think this was supposed to be the Planes, Trains and Automobiles of the new decade. Same basic formula. Howeve
6: Okay, there are worse movies .Not as unbelievable as quot;Pretty Womanquot;, but certainly better acted than anything Juliet Roberts puts out. Seriou
7: here we go, again two gay men meet in a bar and hook up; take drugs; whine a LOT which makes this a G A Y flick. there is supposed to be a subtext ab
8: For Frat Party Afficionados Only Sily to the point of total boredom is a story there this telling is not a story waste your time
9: What a disappointment. I will wait until they release it for viewing on AMERICAN machines, before I go digging around for any special unlock codes to
```

Here are the generated texts. Notice that there are grammatical and punctuation errors. But these are not very significant since we are going to perform additional text processing before text classification.

For class 2.0, here are the original reviews,

```
Here are example of ten sentences from the original text:
0: Felt like propaganda Had intent and pushed the agenda rather than allowing one to draw one's own conclusions. I found this to overshadow the beauty o
1:  star songs  star production This was a terrible production with no life and seemed jumpy and disjointed. It also felt thrown together and not well r
2: Funny Nice set of laughs. Old TV version of medieval subjects are nearly as good as the new ones. Do any of the script writers do research on the ori
3: scared me as a child,but it seams very creaky and tired today in  i saw this movie when it was released and it scared me to death,so when it came out
4: Film Needs A Vial of Billy Bob's LIfe Blood Lara Croft Angelina Jolie is entering a tomb, trying to get to a jewel placed on a pedestal. As she makes
5: Disappointing Incomplete Seasons, no conclusions to multipart episodes. Horribly disappointing. Not complete seasons. Five episodes missing from Seas
6: Not that great I liked the first X-Men, I thought it had great characters, acting, and direction. But X-Men  was just boring. In the first X-Men I re
7: Harrison Ford still kicks butt but Firewall is basically what you've seen too many times before Harrison Ford Hollywood Homicide, Star Wars A New Hop
8: Magic Mike The bodies were good. The dancing was good. The acting was poor. Did not like Matthew McConaughey's role at all. He was a real sleeze. He
9: You Have Be From Mars If You Like This Movie! When this movie first came out I had no interest at all in seeing this movie. It's not that I don't lik
```

Here are the generated texts,

```
0: I will agree with the others I did not get the other  This film is too predictable and over the top. No real insight is given to anything and there i

1: Only for fans of the show The series started out good with the first  episodes, but after  episodes, everything really fell apart. We've seen plenty

2: Boring This was a depressing movie that should have been rated R because it left me with a little bit of a brain fog. I suppose they could have made

3: Not for me I bought this movie because I really like the actors, but the plot was so boring and I didn't care for it. It was like watching a James Bo

4: Good acting but it's not great story line This movie is not a good acting movie at all. I feel the cast wasted their time and money. I hope the peopl

5: Too much hype for little reward When you watch a documentary on the World of Darkness you know that much of what we want to know has been stolen away

6: The story was too simple For a story about love and romance, I expected to see a compelling character develop through some kind of a relationship. In

7: I never knew that this was good until now. I don't know where this movie is headed but if I was in it, I would want to see a book on the basis of one

8: Not bad but not good This movie was an okay movie with the occasional interesting bits. I had a feeling I was watching a movie with two stars, but I

9: Good idea, bad execution I got the idea for this movie from the beginning, so the idea was good.The movie itself was not a great idea. There was some
```

During the training process, we wanted to make sure that the fine-tuned model grasped the tone of the texts properly. But at the same time, we didn't want to fine-tune the model in such a way that it generates very similar texts to the ones in the original text, i.e. we wanted to avoid overfitting. So we trained the model with epoch=2 and epoch=4, numbers such that wouldn't result in overfitting. And then we compared their perplexity on the training text. While the perplexity of the model that was trained with epoch=2 is 45.56, the perplexity of the model that was trained with epoch=4 is 22.95.

Therefore, it is apparent that the model that was trained with epoch=4 creates more realistic texts. So we generated texts with this model.

Additionally, we want to be sure if the fine-tuned model is better than the pre-trained GPT-2 model. On class 1.0 texts, while the vanilla GPT-2's generated texts' perplexity is 46.47, our fine-tuned model's text's perplexity is 22.95.

(All the details can be found under Fine-tuned model's perplexity section of the "MAIN.ipynb" file)

Text Classification:

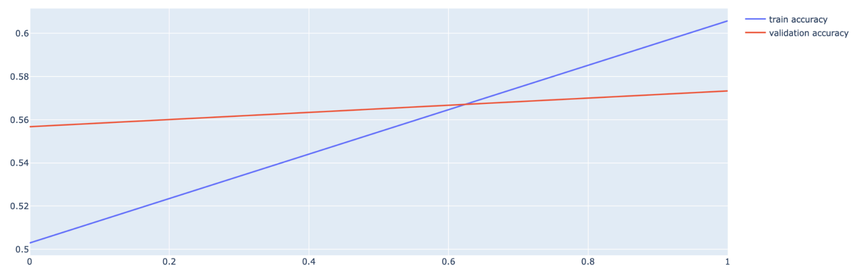
Text Preprocessing for Text Classification:
Before diving into the text classification task, we wondered how text preprocessing has an effect on the accuracy of our model. This is because the raw texts may actually contain information with their weird word capitalization and use of punctuation marks. And we wondered whether the models would be able to take advantage of this or not. To do this, we first fine-tuned the GPT2ForSequenceClassification model with the text that is not preprocessed and with the text that is preprocessed. For this task, by text preprocessing we meant lower-casing, stemming, lemmatization and getting rid of punctuation.

(see “text_processing()” function in the “MAIN.ipynb” file)

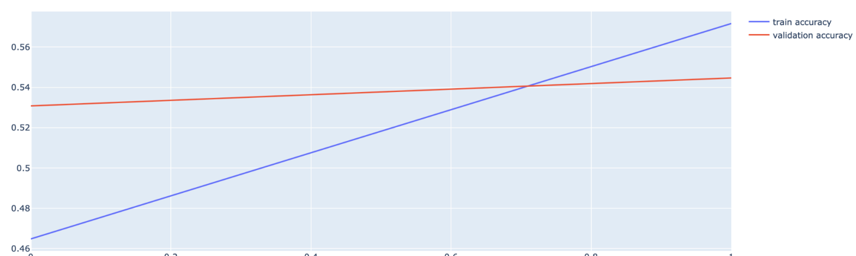
(You can observe the transformation of the text before and after text preprocessing under the Text Processing section in the “MAIN.ipynb” file)

Results:

With epoch=2 and text preprocessing:



Without text preprocessing:



The train and validation accuracies of the model that we trained with preprocessed text are 0.61 and 0.57, respectively. But the train and validation accuracies of the model that we trained without preprocessed text are 0.57 and 0.54, respectively.

Since there is an improvement in accuracy when we do the text classification, we decided to train all the models with the text that is preprocessed.

(Details regarding above findings can be found in the “gpt2_text_classification_notextproc.ipynb” and “gpt2_text_classification_textprocessed.ipynb” files)

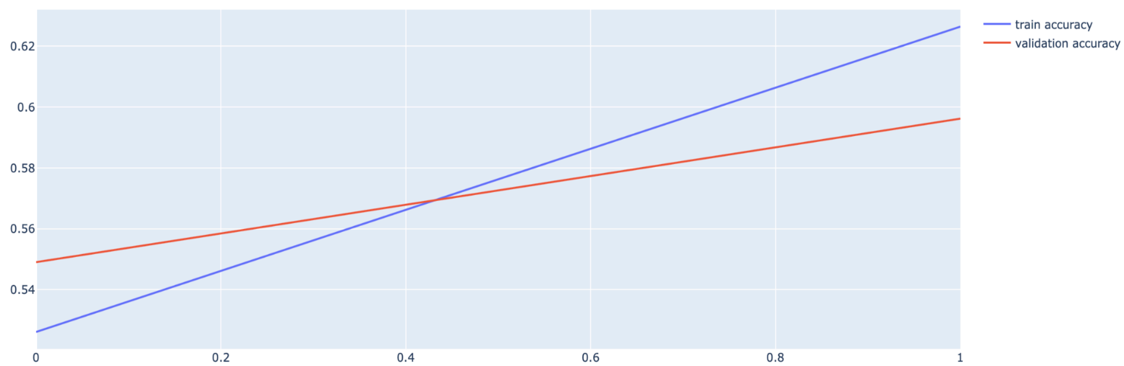
Different Datasets Used in Text Processing:

In order to understand the effect of oversampling, we decided to run each model with two datasets. One dataset is the one that is perfectly balanced in which each class has 10k samples.

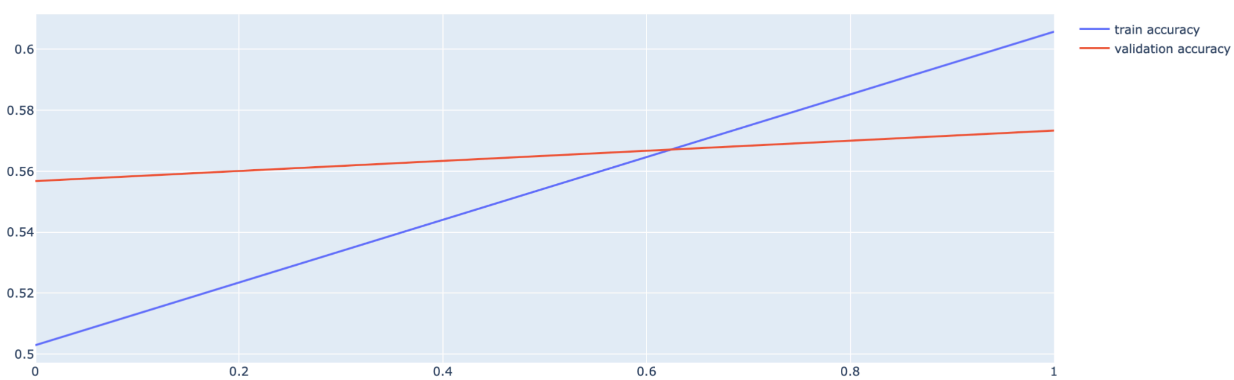
The other dataset is the one that is not perfectly balanced but is close to the first dataset. We decided not to make the second dataset perfectly balanced since otherwise each class has approximately 7k samples. And the accuracy of the models that are trained with less data may be expected to be smaller. To minimize this, we prepared the dataset in such a way that classes 1.0 and 2.0 have their original number of samples and the other classes have 10k samples. From now on we will mention the first dataset as “df_final” and the second dataset as “df_final_oversample”.

Results of GPT-2:

Accuracy of the model trained with “df_final”,



Accuracy of the model trained with “df_final_oversample”,

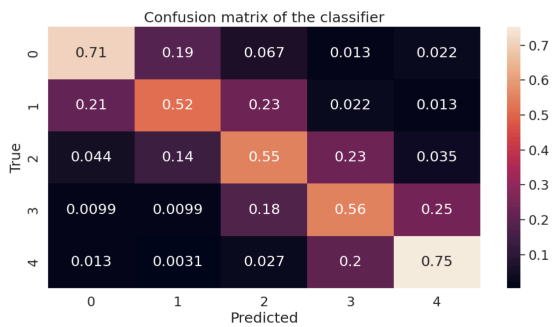


The train and validation accuracy of the model that we trained with “df_final” are 0.61 and 0.57, respectively. But the train and validation accuracy of the model that we trained with “df_final_oversample” are 0.63 and 0.60, respectively.
Therefore, the model that we trained with “df_final_oversample” yielded better accuracy results.

(Details regarding above findings can be found in the “gpt2_text_classification_oversample.ipynb” and “gpt2_text_classification_textprocessed.ipynb” files.)

Results of Roberta:

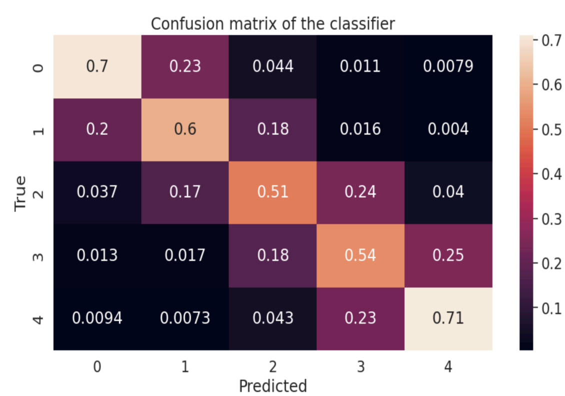
Accuracy of the model trained with “df_final”,




```
flat_accuracy1(predictions,true_labels)
```

```
0.6151779717931498
```

Accuracy of the model trained with “df_final_oversample”,




```
flat_accuracy1(predictions,true_labels)
```

```
0.6112
```

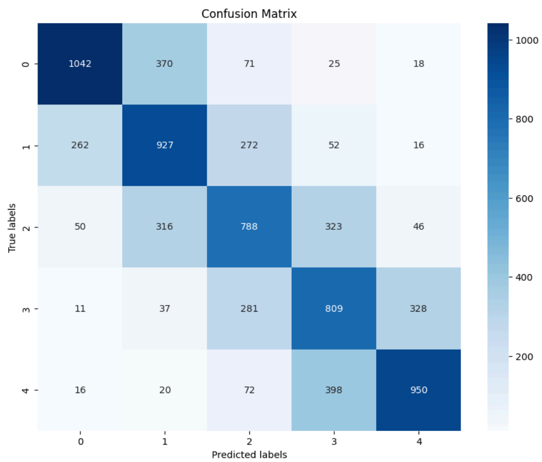
The accuracy of the model that we trained with “df_final” is 0.615. But the accuracy of the model that we trained with “df_final_oversample” is 0.611.

Therefore, both models have very similar accuracies. But from the heatmap, we can observe that the accuracy of class 1.0 remained unchanged but the accuracy of class 2.0 improved from 0.54 to 0.6.

(Details regarding above findings can be found in the “roberta_text_classification_oversample_textproc.ipynb” and “roberta_text_classification_textproc.ipynb” files)

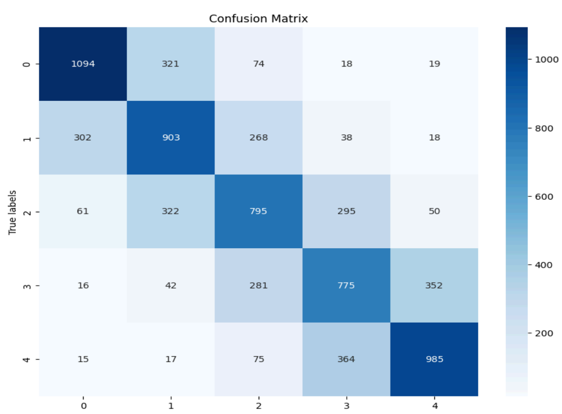
Results of BERT:

Accuracy of the model trained with “df_final”,



Accuracy: 0.6021
Precision: 0.6087
Recall: 0.6021
F1 Score: 0.6044

Accuracy of the model trained with “df_final_oversample”,



Accuracy: 0.6069
Precision: 0.6084
Recall: 0.6069
F1 Score: 0.6075

The accuracy of the model that we trained with “df_final” is 0.602. But the accuracy of the model that we trained with “df_final_oversample” is 0.606.

Therefore, both models have very similar accuracies.
(Details regarding above findings can be found in the “RNN_and_BERT.ipynb” file)

Results of RNN:

Accuracy of the model trained with “df_final”,

```
 50%|█████     | 1/2 [14:14<14:14, 854.16s/it]Epoch [1/2], Loss: 1.7613
100%|██████████| 2/2 [28:26<00:00, 853.12s/it]Epoch [2/2], Loss: 1.7615

Accuracy of the model on the test set: 22.46474143720618%
```

Accuracy of the model trained with “df_final_oversample”,

```
 50%|█████     | 1/2 [16:02<16:02, 962.45s/it]Epoch [1/2], Loss: 1.6132
100%|██████████| 2/2 [32:06<00:00, 963.18s/it]Epoch [2/2], Loss: 1.6123

Accuracy of the model on the test set: 19.83%
```

The accuracy of the model that we trained with “df_final” is 0.22. But the accuracy of the model that we trained with “df_final_oversample” is 0.19.

(Details regarding above findings can be found in the “RNN_and_BERT.ipynb” file)

Conclusion:

Throughout the project we tried to experiment techniques and parameters as possible as we could.

We first experimented if we should fine-tune the GPT-2 for text generation with epoch=2 or epoch=4. From the perplexity results, we decided to train it with epoch=4. We also compared the perplexities of the vanilla GPT-2 and our fine-tuned model. And we concluded that our fine-tuned model generated better texts. Therefore, we oversampled the minority classes with this model.

Secondly, we experimented if we should do text preprocessing when dealing with text classification tasks. To do this, we fine-tuned the GPT-2 model with two texts- one with preprocessed and one without preprocessed. And the accuracy of the model that is fine-tuned with preprocessed text yielded better results. Therefore, we fine-tuned or trained our text classifiers with preprocessed text.

Thirdly, we fine-tuned or trained RNN, Roberta, GPT-2 and BERT with two different datasets- one with perfectly balanced which was oversampled with text generation and the one without oversampled. For most of the models, we got similar accuracy results which were all around 0.61. We can infer two things from this: The generated text doesn't give any relevant information about the text. Or, we should do more text generation. For the first inference, even though the generated texts may not give additional information to the model, it didn't harm the accuracy of the model. So this oversampling technique, to some extent, can be a cure for imbalanced datasets without any harm. For the second inference, we only generated approximately 5k samples for a dataset with a length of approximately 50k. The generated samples are not too much. So it was expected that the effect of this would reflect less on accuracy. But we can see slight improvements in accuracy when we use the oversampled dataset, such as in the fine-tuned GPT-2 model. So this can be a sign that if we generate more texts, we may improve the accuracy. Again, our primary aim was not to boost accuracy. Instead, our main goal was to play around with different techniques and parameters and compare their performance.

Future Work

As discussed above, more efficient models for text generation can be used because the runtime required to sample 2700 samples took about one and a half hours. Secondly, more NN models can be used, such as LSTM, Bidirectional LSTM and FFNN to make a better comparison of different NN models. Thirdly, different embedding techniques can be used for RNN such as GloVe and Word2vec. As can be seen in the "RNN_and_BERT.ipnyb" file, we tried to use Word2vec embedding. But it took 2 hours+ to just do the embedding . So for future work, we could do more text processing, like removing stop words to shorten the text, to create faster embedding matrices. Lastly, more texts can be generated in order to better understand the effect of this oversampling technique on accuracy.

Shared Responsibilities:

Can Erozer:

- Fine-tuned GPT-2 models and generated texts with different parameters
- Calculated the perplexity of the models and decided which models to use
- Determined the need of text processing by comparing it on a model
- Fine-tuned Roberta and GPT-2 and analyzed their results
- Wrote the decision flow, conclusion , experiment setup, and future work sections

Jialu Li:

- Worked with text preprocessing tools
- Implemented RNN. Worked with word2vec embeddings and tf-idf. Evaluated its results
- Fine-tuned BERT and evaluated the results
- Analyzed the findings from the models
- Wrote the abstract section