Step 1: Choose an NLP problem

Step 2: Identify or construct a solution based on a generative probabilistic (language) model. Describe the model in detail and develop a solution using parameter inference (and/or decoding).

Step 3: Identify or construct a solution based on a (discriminative) neural network. Describe the network structure in detail and develop a solution using gradient descent.

Step 4: Train and apply both approaches to "real" data acquired legally. Evaluate the results qualitatively and quantitatively. Highlight situations where each approach performs well and poorly. Any unusual/unexpected results require explanation (and frankly, probably debugging).

Step 5: Train and apply both approaches to synthetic data that you generate according to the generative model from Step 2. Evaluate the results qualitatively and quantitatively. Highlight situations where each approach performs well and poorly.

Step 6: Discuss pros and cons of the two approaches. Consider: • quality/correctness • data, time, and computational requirements • interpretability

Dataset:
https://www.kaggle.com/datasets/ilhamfp31/yelp-review-dataset

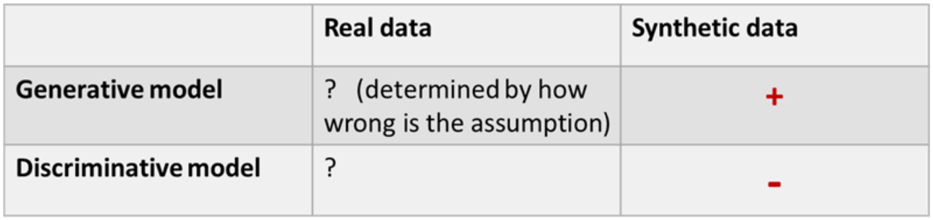
10/26 Recording Patrick解释了**final project:**

0. (done) Preliminary EDA
1. (done) train generative model on real data
2. (done) generate synthetic data(size should > real data) by generative model
    a. Use probability in Naive bayes to generate synthetic data, sentence don't have to make sense, just word following word based on probability
    b. Ref: https://github.com/wh153/Sentiment-Analysis-of-Human-Reviews-/blob/main/nb.py
    Another ref: https://github.com/unsupervisedlearner1123/703-final-project/blob/main/10_code/10_nb_sentiment140clean.py
    或者他可以直接看naive bayes的原理自己写！原理: (https://drive.google.com/file/d/1vqiWW9motR0hTZ9joMH23dnDkokABoxB/view?usp=sharing)
3. (done) train generative model on the synthetic data you generated
4. (done) train discriminative model on synthetic data
5. (done) train discriminative model on real data (can be done at any step)

Results should be like:

Whatever metric we use, generative model on synthetic data should be better than discriminative model on synthetic data (if not then you probably do it wrong)

On real data, discriminative model (应该大概率) better than generative model



	Real data	Synthetic data
Generative model	? (determined by how wrong is the assumption)	+
Discriminative model	?	-

Plots in report:

1. **Four confusion matrix**

	Real data	Synthetic data
NB		
LSTM		

2. **Acc & loss function curve**

Report ref:
https://github.com/razalamb1/political_speech
https://github.com/wh153/Sentiment-Analysis-of-Human-Reviews-

Sentiment Analysis on Yelp Reviews

Jiaxin Ying, Ruixin Lou, Yuanjing Zhu

1. Introduction

In this project, we are interested in the topic of sentiment analysis, which is an approach in natural language processing that identifies the emotional tone behind a body of text. In practical application, sentiment analysis tools are essential to detect and understand customer feelings. It is an important factor when it comes to product and brand recognition, customer loyalty, customer satisfaction, advertising and promotion's success, and product acceptance. Hence, we chose the dataset from Kaggle, which consists of reviews from Yelp in 2015, and we will apply Naïve Bayes model and LSTM (Long Short-Term Memory) on this data and analyze the results.

2. Methodology

2.1 Data

Word cloud of Positive yelp reviews
Word cloud of Negative yelp reviews
place
food
great
love
good
best
time
servic
first
worst
review

Fig. 1. Word cloud of Yelp reviews

Our Yelp reviews dataset was obtained from Kaggle and is made up of Yelp reviews [1]. It was taken from the 2015 Yelp Dataset Challenge data information. The Yelp review polarity dataset is created by taking into account the analysis of positive and negative reviews. There are 38,000 testing samples and 560,000 training samples combined in total in Kaggle. Class 1 polarity is negative, and class 2 polarity is positive.

Because the train.csv is large, we mainly use the test.csv file to analyze Yelp reviews and found that it's large enough for training models. It contains two columns, which refers to class index (1 and 2) and review text. Double quotes (") are used to escape the review paragraphs, and any inside double quotes are separated by two double quotes (""). A backslash and a "n" letter, which together form the symbol "\n," are used to separate new lines.

2.2 Generative model – Naive Bayes

Naïve Bayes [2] (NB) is one of the simplest generative probabilistic models to solve the classification problem. The foundation behind NB is the Bayes' rule:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)},$$

where $P(y)$ is the conditional probability of an event x happening given that another event y has already occurred. In our problem, we would like to classify Yelp reviews as negative or positive ones, labeled as 1 and 2 respectively. Then how should we treat or encode the text? A method called bag-of-words (BoW) is adopted. A BoW is a representation of text that describes the occurrence of words within a document, ignoring their order. For example, a piece of reviews reads:

"The food is good. Unfortunately, the service is very hit or miss. The main issue seems to be with the kitchen, the waiters and waitresses are often very apologetic for the long waits and it is pretty obvious that some of them avoid the tables after taking the initial order to avoid hearing complaints".

Immediately, we notice that ***the*** occurs 8 times, ***and*** occurs 2 times, ***very*** occurs 2 times and so on. The BoW looks like

the	8
is	2
and	2
avoid	2
very	2
to	2
unfortunately	1
⋮	⋮

Obviously, some words are useless, such as ***the***, ***is***, and ***and***. On the other hand, the word ***avoid*** appears twice and ***unfortunately*** shows up once. We may consider it as a negative review. In fact, it is. After preprocessing the raw reviews, most of meaningless words (stopwords) will be removed from documents, and only useful words will be kept. For a document d, our best estimate of the correct class $\hat{c}$ is

$$\hat{c} = \mathrm{arg}\, arg \max_{c \in C} P(c|d) .$$

By the Bayes' rule and the assumption of independency of features of each documents, we can have

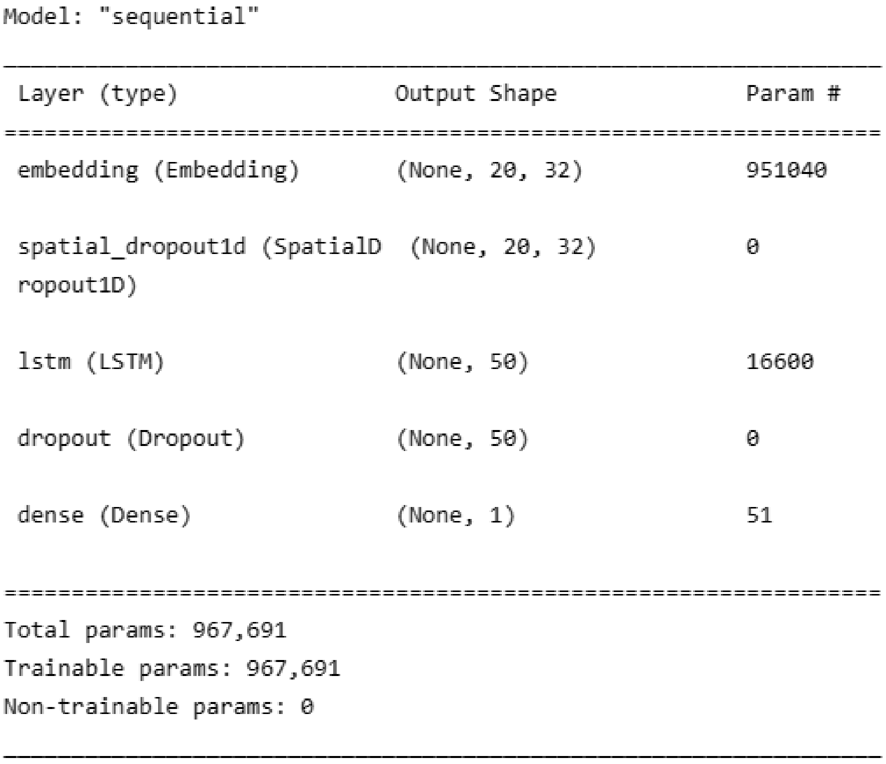
$$\hat{c} = \mathrm{arg}\, arg \max_{c \in C} \mathrm{log}\, log\, P(c) + \sum_{i \in positions}^{n} \mathrm{log}\, log\, P\left(w_i|c\right),$$

where w_i are different words in the document.

Before training the NB model, we first preprocess the data (original reviews from Yelp). We change all words in lowercase, remove all punctuations and special characters as well as stop words such as "the", "a", and "is". Besides, stemming words also plays an important role in our preprocessing. To train and test the model, we first split the preprocessed dataset into training and testing datasets with an 80-20 split. There's no need to reinvent the wheel when it works and is efficient. We also use the CountVectorizer() class to convert the collection of text documents into a matrix of token counts. Therefore, we use multinomial Naive Bayes model provided by Scikit-learn to classify the documents with additive (Laplace/Lidstone) smoothing parameter equal to one. The result of the NB model on real and synthetic data is presented and discussed in Section 3.

2.3 Discriminative model – LSTM [3]

Our neural network model is based on LSTM which is an efficient way to do sentiment analysis since it is able to remember and utilize information from long-term dependencies in the input data. We use Tensorflow to realize this purpose. For the neural network, the input needs to be numeric. We first initialize the tokenizer with a 5000 word limit, which is the number of words we would like to encode. Then we create associations of words and numbers, and replace the words in a sentence with their respective associated numbers. Since the length of each document is different and LSTM requires inputs to have equal lengths, we have to add the sequence to have the chosen length of inputs (20 in our model). The structure and parameters of LSTM is as shown below.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 32)            951040

 spatial_dropout1d (SpatialD  (None, 20, 32)           0
 ropout1D)

 lstm (LSTM)                 (None, 50)                16600

 dropout (Dropout)           (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 967,691
Trainable params: 967,691
Non-trainable params: 0
_________________________________________________________________
```

Fig. 2. LSTM architecture

Our neural network contains 5 layers. The first layer is an embedding layer, which converts the input sequences into dense vectors of a fixed size, called embedding vectors. The length of the embedding vector is specified as 32 and input length is 20. The second layer is a spatial dropout layer, which randomly sets 25% of input units to 0 at each update during training time to help prevent overfitting. The third layer is an LSTM layer with 50 units, which is a type of recurrent neural network that can process long sequences of data. The dropout percent and recurrent drop-out percent are both set to be 50. The fourth layer is a dropout layer, which randomly sets 20 percent of input units to 0 at each update. The final layer is a dense layer with sigmoid function as activation function. Since our problem is a binary classification problem, binary cross entropy is suitable as loss function and Adam is used as an optimizer. To train the model, we found that 10 epochs and a batch size of 32 are enough to generate a stable result. The result of the LSTM model on real and synthetic data is presented and discussed in Section 3 as well.

2.4 Synthetic data generation [4]

One reason for calling Naïve Bayes a generative model is that NB can generate synthetic data. When we have a NB model, it means that we have estimates of $P(c)$ and $P\left(w_i|c\right)$, which we can use to generate documents. We have to generate sentences word by word. First, we sample a class from $P(c)$. Next, we keep generating words by sampling from $P(d|c) = \prod_{i=1}^{n} P\left(w_i|c\right)$. One possible problem is that we may not know when to stop. We can either enforce the length of the generated sentences is less than a finite number l, or add $\#S$ and $\#E$ to the begin and end of the original documents as words, then if we observe an ordered pair of $\#S$ and $\#E$ in the process of generating data, we can slice the words between them to form a new sentence. In practice, we combine two methods together. One obvious drawback of the synthetic data is that the sentences are very likely to make no sense, since NB ignores the relationship between words.

3. Results and Discussion

3.1 Synthetic data results

After creating the synthetic dataset, we used it to construct a naïve bayes classifier and an LSTM neural network. We split the dataset so that the training set contains 80% of the data and the test set contains 20%. Figure 3 displays the confusion matrix of the LSTM and naive bayes models on the test set.

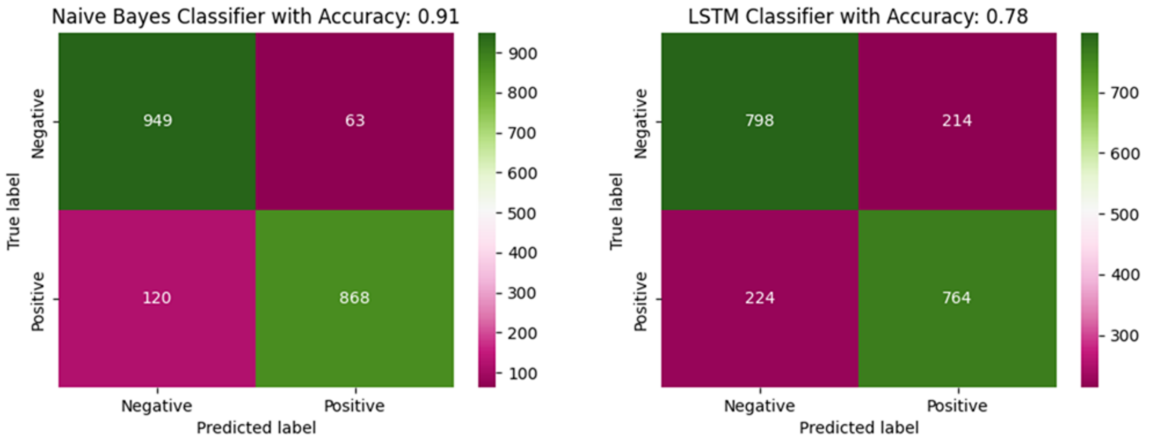


Fig. 3. Confusion matrix of naive bayes and LSTM classifier on synthetic data

Naive bayes classifier achieved an accuracy of 91% while the accuracy of LSTM is 78%. Both classifiers are more effective at identifying negative reviews, but they fail to classify short and ambiguous sentences like *"oh yeah", "review often"*, etc. Since the synthetic data is generated by naive bayes, it makes sense that naive bayes outperforms LSTM because the synthetic data was created specifically to fit the assumptions of the naive bayes algorithm. Because the synthetic data was generated using the same assumptions as the naive bayes model, naive bayes model can accurately predict the sentiment of the text.

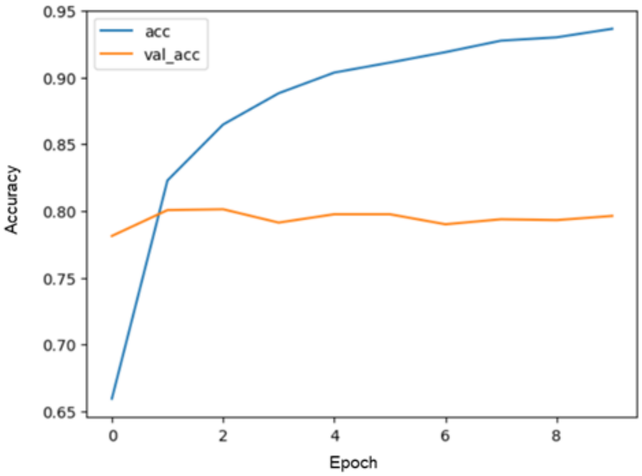


Fig. 4. Training process of LSTM on synthetic data

Figure 4 shows the accuracy curve of training and test sets of LSTM in each epoch. The accuracy of the training set is much higher than the test set, indicating the issue of overfitting. The reasons why the LSTM model overfit may due to ①: LSTM is too complex for the amount of synthetic data: LSTM models can have a lot of parameters, and if there is not enough training data, the model may learn patterns in the training data that do not generalize to new data; ②: Regularization parameters are not optimal: we set the "dropout" parameter to 0.5 and "recurrent_dropout" parameter to 0.5 within LSTM layer as well as adding a "Dropout (0.2)" layer, but these hyperparameters need to be tuned further to better prevent overfitting.

3.2 Real data results

When applying naive bayes and LSTM on real-life dataset, we continue to divide the training and test sets in an 8:2 ratio and maintain that every model parameter is the same as the one we created on synthetic data. Figure 5 displays the confusion matrix generated from two models.

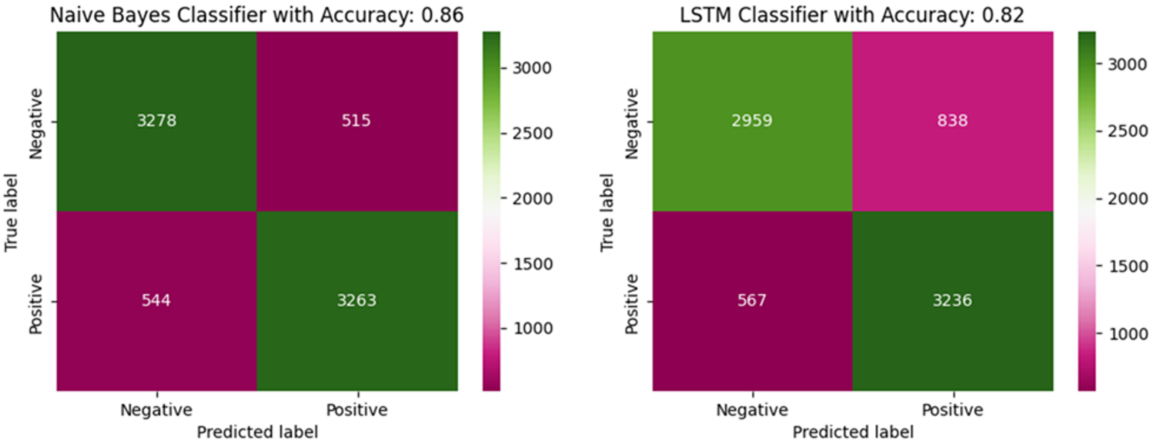


Fig. 5. Confusion matrix of naive bayes and LSTM classifier on real data

On real data, naive bayes continuously outperforms LSTM, obtaining 86% accuracy compared to 82% for the LSTM model. The reason why naive bayes model is more effective than the LSTM network is probably due to the fact that naive bayes is better suited to this dataset in this task. The assumption of naive bayes when doing sentiment analysis is that the features (i.e. words) in the input text are independent of one another. In other words, the presence or absence of a particular feature does not depend on the presence or absence of any other feature. By examining the words that are most frequently found in positive/negative category, we found that positive reviews often contain words like “great”, “good”, “really”, “time” while the negative review are more likely to include the words “never”, “bad”, “little”, “wait”. Therefore, using naive bayes can be effective at detecting these words and thus selecting the appropriate sentiment as a result. Some cases when naive bayes perform well include “*Great location, Front Desk Staff were very personable ……”, “Very bad experience. The food was below average ……*”, but it still has trouble identifying reviews which contain both compliment and complaining content.

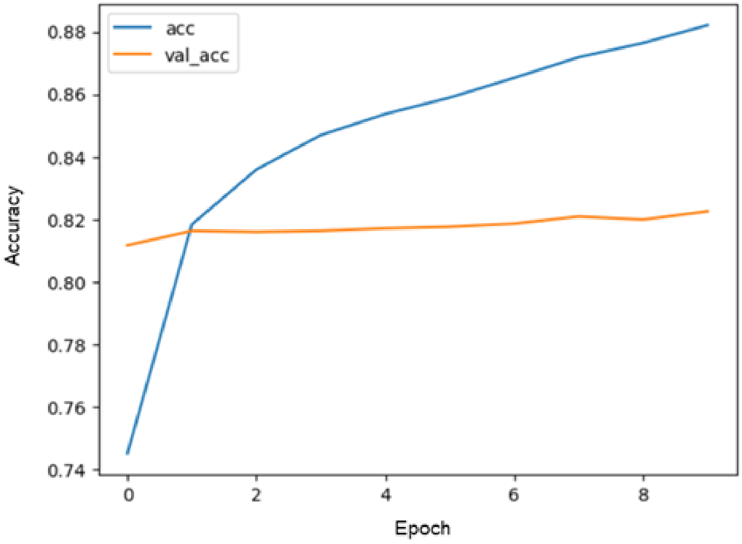


Fig. 6. Training process of LSTM on real data

The LSTM network's training procedure is shown in Figure 6. It still has the overfitting issue, which may be another factor contributing to its worse performance compared to naive bayes. We also note the issue that the LSTM classifier struggles to recognize negative reviews. Therefore, we examined the negative reviews that LSTM mistook for positive ones, and the following are some cases:

“Some good some bad. This goes for the food and the service ……”
“I’ve been here quite a few times, and I always like it cause …… But this time when I ……”
“I could visit this place four times and give it four different reviews …….”
“Why would anyone pay money for food like this?”

We can see that some reviews are written in a way that is difficult to understand, making it hard for the LSTM model to accurately classify them. This could be due to ambiguity, sarcasm, adversative transitions, use of slang or colloquial language, and other factors.

3.3 Comparison

Table 1 compares the correctness, computational time, computational requirements, and interpretability of naive bayes versus LSTM. Although naive Bayes and LSTM can both be useful for sentiment analysis, for this dataset and the current sentiment analysis task, naive Bayes appears to be the superior choice.

Table 1. Comparison between naive bayes and LSTM

	Naive Bayes		LSTM	
Correctness	91% (s)	86% (r)	78% (s)	82% (r)
Computational time	0.7s (s)	0.1s (r)	34s (s)	2.4min (r)
Computational requirements	CPU		GPU required	
Interpretability	More interpretable		Hart to interpret (black box)	

s: synthetic data, r: real data

Naive Bayes classifier is a simple and fast algorithm, making it efficient for large datasets and requires little data to make predictions. It can be trained only using CPU and the computational time for training and testing is really small, usually within 1 second. Naive Bayes is also generally more interpretable compared to neural networks, as it provides clear probabilities for each class and feature. In terms of correctness, it relies heavily on the assumptions of naive bayes algorithms, which is the independence between features. Without doubt the accuracy of naive bayes classifier on synthetic data is the highest among the four since the synthetic data is generated based on the assumption of naive bayes. However the assumption might be violated in real-world data, leading to poor model performance, which is not the case in our analysis.

Generally LSTM may perform better due to its ability to capture long-term dependencies in the data allowing it to learn complex patterns and relationships, but the performance of LSTM can also be sensitive to the quality and size of the training data, and it may not perform well if the data is noisy or limited. LSTM requires a large amount of data and computing power (e.g. GPU) to train, making it resource-intensive. The computational time of LSTM is hundreds of times longer than that of naive bayes classifier and tuning LSTM network to its optimal model performance requires time and effort. In addition, LSTM is prone to overfitting, especially if the dataset is small or has a limited number of features. That’s the reason why the LSTM classifier on synthetic data achieved the lowest accuracy. It can also be difficult to interpret and understand the internal workings of an LSTM model, making it less transparent compared to other machine learning algorithms (e.g. naive bayes).

Overall, the choice between naive Bayes and LSTM for sentiment analysis will depend on the specific needs and constraints of the task, such as the amount and quality of the data, the required accuracy, and the available resources.

4. Conclusion and Limitation

4.1 Conclusion

In this project, we gathered data from kaggle and preprocessed it by making all the letters lowercase and removing punctuation, numbers, special characters, and so on. Then Naive Bayes model was applied to generate synthetic data. We also built a LSTM and made a comparison model between them. In conclusion, the Naïve Bayes model has better performance on both the real data and synthetic data than the LSTM model. In comparison to the LSTM, Naïve Bayes has several advantages, and the first is efficiency. It requires less computation time, and we do not need a GPU to train the model, which is a requirement for LSTM. In addition, Naïve Bayes has better interpretability than LSTM.

4.2 Limitation

However, there are some limitations to our study. One thing is that the conclusion we draw is only for this dataset and task. The assumption of Naïve Bayes does not always hold. Real-world data often contains more variables and interactions which can violate the assumption of Naïve Bayes, leading to lower accuracy. Therefore, it is important to test the performance of Naïve Bayes on specific data to get an understanding of its capability. Moreover, the sample size of the dataset is too small for LSTM, so the model has the problem of overfitting. Hyper-parameter tuning is needed for future improvement.

Reference

[1] Yelp Review Data. https://www.kaggle.com/datasets/ilhamfp31/yelp-review-dataset
[2] Jurafsky, Dan and James H. Martin. "Speech and Language Processing." (2000).
[3] Deep Learning. Available:
https://medium.datadriveninvestor.com/deep-learning-lstm-for-sentiment-analysis-in-tensorflow-with-keras-api-92e62cde7626
[4] Sentiment Analysis. https://github.com/rw417/Sentiment-Analysis-of-Human-Reviews-

Check out our GitHub repo: https://github.com/nogibjj/NLP_yz_jy_rl