Amazon Review Sentiment Analysis

EE 660 Course Project

Project Type: Design a system based on real-world data

Number of student authors: 1

Zixi Liu zixiliu@usc.edu

12.08.2019

1. Abstract

Amazon Review Sentiment Analysis is a natural language processing challenge that aims at predicting Amazon customers' rating from their reviews. This challenge is a classical natural language understanding task, because reviews are proportional to ratings, both of which reflect satisfactory of customers. In this project, I am building a language processing system to fit the relation between these two highly correlated pair and the main challenge is dealing with highly unbalanced classes and find out a proper presentation for each sentence. In this paper, I will explore several techniques in solving this problem, including TF-IDF and neural word embedding for feature engineering, SMOTE for re-sampling and logistic regression, linear regression, decision tree, Adaboost for modeling. The model finally achieve a macro score as 0.365, a 66.7% improvement compared with baseline (0.219 in macro score) which use a ridge regression classifier with TF-IDF.

2. Introduction

2.1. Problem Type, Statement and Goals

Text sentiment analysis is an important task for a large amount of problems related to human-computer interaction and language understanding and modeling. This task plays an important role in a plenty of fields such as sociology, marketing and advertising, psychology, economics, and political science. In this project, amazon review sentiment analysis is a 5-class classification task that predicts customers rating (in 5 level) for product by their reviews. The inherent nature of Amazon reviews is a complicated and poses many challenges for traditional analysis methods. Firstly, some of these challenges are from its categorical nature. The nature of customers' reviews are human language and human language has a very random nature, even though there are some structures and rules in it. This categorical nature push the feature space to a very high dimentionality. Traditional models are mainly dealing with continuous numerical features in a moderate dimentionality. Secondly, the sentiment and the semantic meaning in a review is very hard to measure. To solve this challenge, we need to find a good feature that can provide some probabilistic nature of the sentiment measure and semantic measure in the language. Thirdly, some challenges are from highly imbalance classes in this problem. As we can see in review website, the reviews with 1 star is usually much fewer than those with 5 stars, but these minor classes reviews are usually more important for manufacturers . Therefore, how to let the model pay more attention to the minor classes is also a big difficulty in this project.

2.2. Literature Review

As a traditional task in natural language processing research field, there are plenty of literature and approach existed to solve this problem. It can roughly divide into two period. The first period is before 2012 when researchers in this fields mainly use artificial featuring engineering to solve this problem. The second period is after 2013 when unsupervised feature learning techniques like word2vec [1] and GloVe [2] became popular and provide substantial improvement for downstream task in tremendous data learning scenario using deep learning technique. Here I provide a sample machine learning pipeline [3] before 2012 as picture below. This project has a similar pipeline as below. One thing different is that I will try to use both word2vec and TF-IDF techniques and compare the performance of these features with the same downstream task.

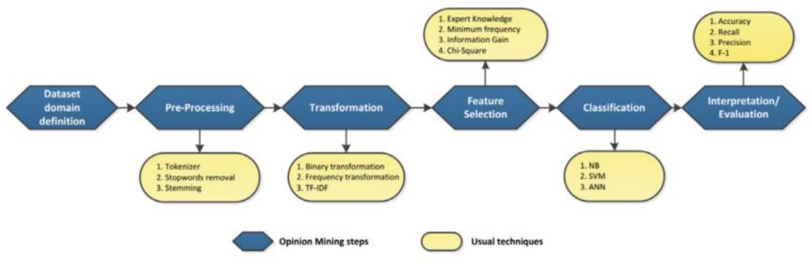


Fig. 1 Text sentiment Analysis Pipeline [1]

2.3. Our Prior and Related Work

This project doesn't have any directly prior work, though there are some works related to this. In this semester, I am taking CSCI 544 natural language processing and I learn some basic techniques about language processing, including TF-IDF and word2vec. However, this class mainly focus on using deep learning techniques to solve some leading edge challenge in research, but this project is actually focusing on using traditional machine learning technique to solve the problem.

2.4. Overview of Our Approach

In this project, I use two different language models as feature extraction methods which are TF-IDF and word2vec. Among them, word2vec model is to learn semantic vectors of words by using an unsupervised machine learning model based on 2-layer perceptron classification machine. TF-IDF model is term frequency counting model and I will use PCA algorithm to reduce feature dimension. In order to deal with class imbalance, I use SMOTE techniques to do re-sampling. In the classification, I fine tune and compare the performance between logistic regression, linear regression, decision tree, Adaboost and Gaussian Naive Bayes. In some of this technique, I also use regularization method to do feature reduction. Finally, the evaluation is mainly use F1-macro score for overall performance comparison and F1 score for comparing the performance in each class.

3. Implementation

3.1. Data Set

The dataset I use in this project is from Kaggle which is a smaller sample of Amazon Customer Review Dataset [4] and full dataset is available in Datafinti's Product Dataset [5]. For sentiment analysis, I use the column of reviews.text as input and reviews.rating as output.

In this section, a statistic about the basic information of this dataset will be shown, and some corresponding preprocessing pipeline will be built in next section. This section is referred from [6]. The full dataset columns, their data types and their number of missing data are shown in Chart 1. The reviews.text column is strings and reviews.rating column is integer with 5 categories from 1 to 5.

Though only one input column is used here but since they are strings, the data nature is actually categorical. Examples for 5 categories reviews are shown in Chart 2. As we can see in the sample reviews, there are a lot of redundant punctuation and some words in the sentence are also wrong. So, in the 3.2 section, a text cleaning pipeline will be built there. As we can see in Chart 1, the missing data in review.text and review.rating is added up to be 34, which is a small number compared with the complete instances we have, so these incomplete data will be discarded. Fig. 1[] is the statistic information of data set. Review rating shows that we have great class imbalance in this data set.

total num of instance: 34660, num of column: 21					
Columns:	data type	num of missing data	column	data type	num of missing data
id	string	0	reviews.doRecommend	Boolean	594
name	string	6760	reviews.id	int	34659
asins	string	2	reviews.numHelpful	int	529
brand	string	0	reviews.rating	int	33
categories	string	0	reviews.sourceURLs	string	0
keys	string	0	reviews.text	string	1
manufacturer	string	0	reviews.title	string	5
reviews.date	time	39	reviews.userCity	string	34660
reviews.dateAdded	time	10621	reviews.userProvince	string	34660
reviews.dateSeen	string	0	reviews.username	string	2
reviews.didPurchase	boolean	34659			

Chart 1. Basic statistic of dataset

Reviews	Rating
Love love love my kindle fire 8.....this is what my 9 yr old granddaughter said when I bought this for her at Christmas..we have purchased kindled in the past but this one has been the best....love the fact that you can now use a memory card in it...that helps so much when you use it for books...games...music and viedos...thanks,Kindle fire 8,,,Mynie	5
I purchased this when my last tablet died. It meets all basic needs and the price was great.,Great price for a basic tablet.,,,doglover	4
I was hoping to use Google launcher with this tablet but it is really locked down and you cannot change the launcher or the lock screen. Still cheap and fine for watching movies.,A cheap tablet,,,blargh22	3

Didn't have some of the features I was looking for. Returned it the next day. May be good for others,Wasn't for me,,,Mark	2
not good, hate it , never buy it again, sucks. done	1

Chart 2. Examples of review text

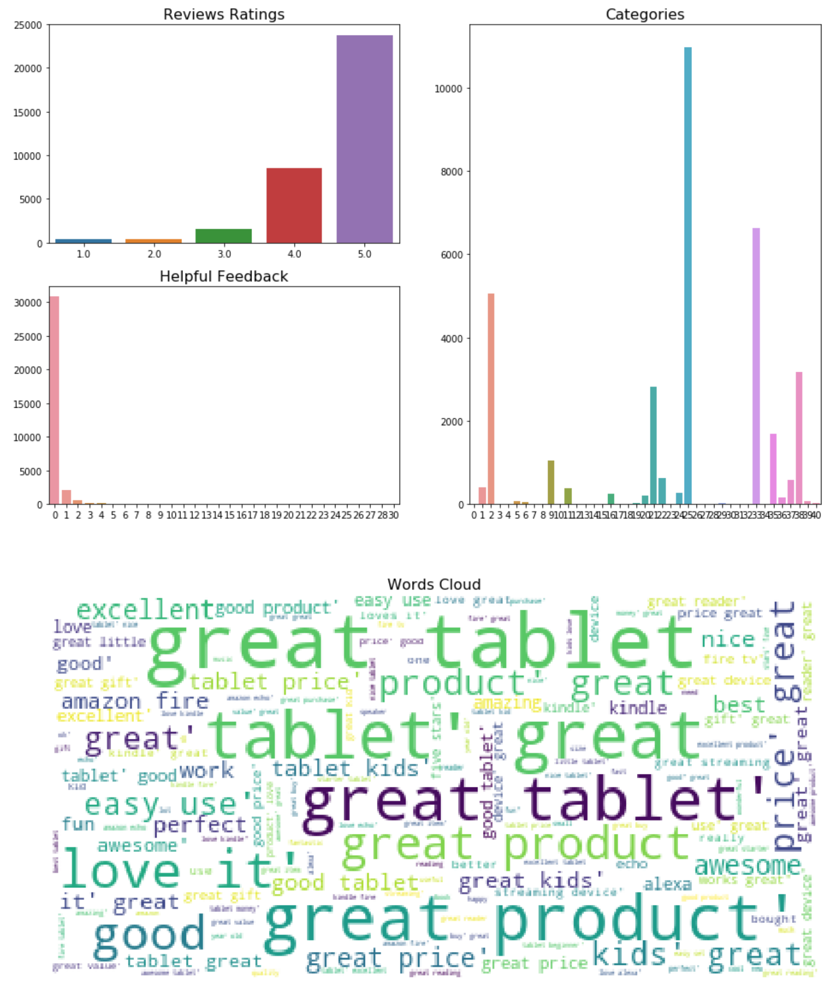


Fig. 1

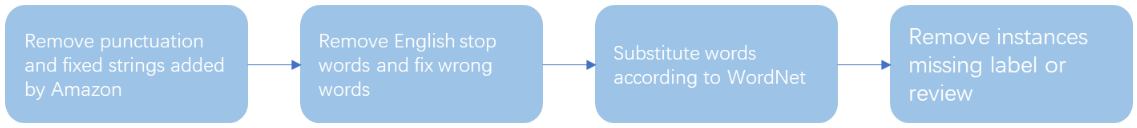
3.2. Preprocessing, Feature Extraction, Dimensionality Adjustment

In this section, I will introduce data preprocessing pipeline, two feature extraction methods and PCA based dimensionality adjustment.

3.2.1 Denoising

As we can see in 3.1 section, the review text is noisy and there are lots of words that repeat plenty of times but have little use with target task, like “the”, “them”, “I”, “there”, etc. In NLP realm, these words are called stop words, which does not help on most of NLP problem such as semantic analysis, classification, etc. So, I will use a `stopwords` API from NLTK library to remove these words. Also, in English, words will have different form, like singular and plural, so, I also use `WordNetLemmatizer` API from NLTK to reduce data sparsity.

Fig. 2 Data Preprocessing Pipeline


Remove punctuation and fixed strings added by Amazon
Remove English stop words and fix wrong words
Substitute words according to WordNet
Remove instances missing label or review

3.2.2 Feature Extraction

In this project, I will use two different kind of features for comparison. So after the feature extraction, two machine learning pipeline will be ran parallelly.

3.2.2.1 TF-IDF feature extraction

TF-IDF means Term Frequency-Inverse Document Frequency (TF-IDF) vector, which is a sentence vector and each feature in it is the normalized term frequency of a word in corpus. TF-IDF is a commonly used feature in NLP society and I use `TfidfVectorizer` API [7] from sciket learn library to implement this algorithm. Here to avoid data snooping, I fit the TF-IDF model only in training set and use this model to get TF-IDF vector for each sentence in validation set and test set.

Since the feature number in TF-IDF vector depends on the corpus counting in the training set, the feature number will be very large. In this project, the feature number goes to 9676, but we only have 22159 training instances. So, to avoid curse of dimensionality, I implement PCA for feature reduction and the number of components to keep is the component whose variance is over 95% of the other features and finally I reduce feature dimension to 3192. Again, PCA is only being done on the train set and vectors in validation set and test set are only projected onto the direction determine by training set. In addition, normally speaking, PCA should be done after the data is normalized, but I will not do normalized for TF-IDF vectors because TF-IDF vector has been normalized in the algorithm. The following two pictures show that normalized TF-IDF feature before PCA will harm the feature.

3.2.2.2 Word2Vector Feature Extraction and SMOTE

Word2Vector is a word feature learned by unsupervised learning through multi-layer perceptron. The training architecture is shown as the Fig. 3 below.

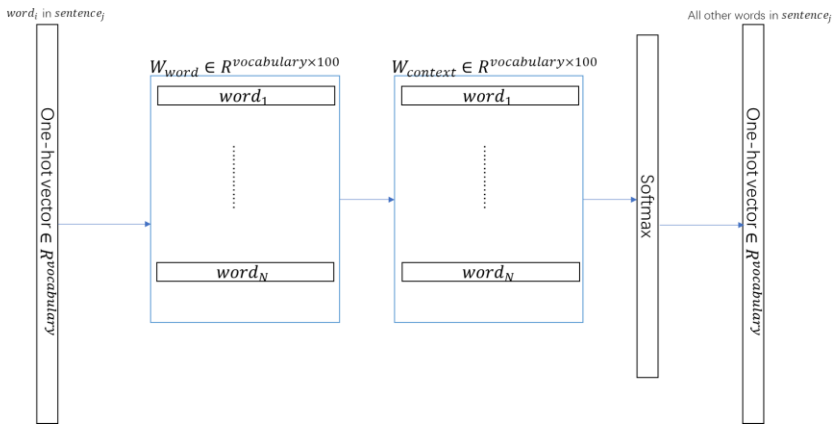

$word_i$ in $sentence_j$
One-hot vector $\in R^{vocabulary}$
$W_{word} \in R^{vocabulary \times 100}$
$word_1$
$word_N$
$W_{context} \in R^{vocabulary \times 100}$
$word_1$
$word_N$
Softmax
All other words in $sentence_j$
One-hot vector $\in R^{vocabulary}$

Fig. 3 Word2Vec training architecture

This algorithm use dot product between different words to evaluate similarity between words in a sentence. Therefore, every word will project into a continuous dense space where words with similar semantic meaning are clustered together.

With Word2Vec feature, a sentence can be represented in a matrix where each row is a vector for a word in the sentence and the row number is the sentence length. Sentence vector is calculated by taking average on word vectors. I use APIs from gensim library [8] to generate a 50 dimension Word2Vec feature.

Finally, I use 10-nearest neighbor SMOTE algorithm to compensate for unbalance classes in training data set. Default SMOTE algorithm resample all minor classes to the number of majority class. In the train set, data number before and after SMOTE is shown in Chart 3.

class	5	4	3	2	1
number before SMOTE	15214	5466	959	257	263
ratio before SMOTE	0.68	0.25	0.04	0.01	0.01
number after SMOTE	15214	10000	2000	1000	1000
ratio after SMOTE	0.52	0.34	0.068	0.034	0.034

Chart. 3 SMOTE on Word2Vec feature

3.3. Dataset Methodology

The data set is used as the following flow chart.

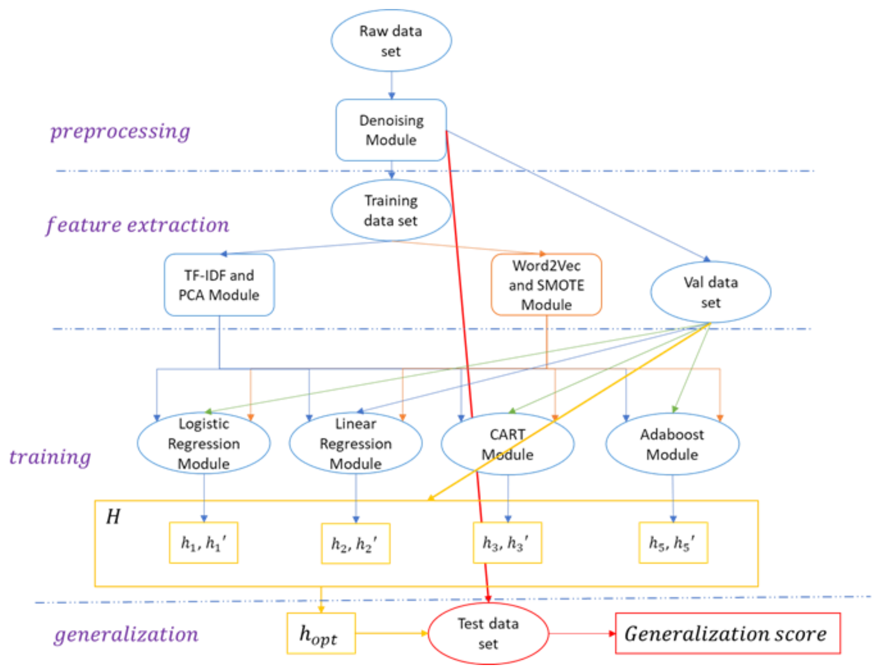


Fig. 4 Machine Learning Flow Chart

The whole data set is firstly passed through a denoising module as mentioned in 3.2.2.1 and then whole data set is separated into training set, validation set and test set randomly but keeping the class proportion the same in each class. In the feature extraction module, only training set is used to fit the unsupervised learning model. Using these two features, two parallel machine learning flows are running as Fig. 4. h_i means the optimal hypothesis in a specific model using TF-IDF feature selected by validation set. h_i' means the optimal hypothesis in a specific model using Word2Vec feature selected by validation set. Final optimal hypothesis is selected under all the 8 sub-optimal hypothesis using validation set. Test data is used to get a generalization score.

3.4. Training Process

This project is an unbalanced multiclass classification task, so in order to have a balanced evaluation for both minor class and majority class, I use F1-macro score for model selection. The following are the training procedure for each model shown in Fig. 4.

Multinomial Logistic Regression:

- Multinomial Logistic Regression is modeling as blow:

$$P(Y_i = l | \boldsymbol{x} = \boldsymbol{x_i}) = \frac{\exp(\beta_l \boldsymbol{x_i})}{\sum_{l'=1}^{5} \exp(\beta_{l'} \boldsymbol{x_i})}$$

$where\ l\ is\ class\ index.$

Since the data set have very unbalanced data, I choose a weighted version of loss function, that is weighted cross-entropy.

$$J(\vec{\theta}) = -[\sum_{i=i}^{N}\sum_{l=1}^{5} w_l 1\{Y_i = l\} \log\ P(Y_i = l|\boldsymbol{x_i}; \vec{\theta}]$$

$$where\ w_l = \frac{number\ of\ data\ points\ of\ class\ l}{total\ data\ point}$$

To solve this loss function, I use SAGA solver in sciket learn which is a improvement of Stochastic Average Gradient solver. All the model and loss function are implemented using LogisticRegression in Sciket Learn library. Implementing detail can be found in my source code.

- Logistic regression provides a flexible boundary for classification. In project, I assume that the problem should be a non-linear problem, because we can intuitively feel that we don't express our emotion in a monotonic way.
- There are two hyperparameters need to be chosen in this problem. One is the choice of penalty which has three choices L1, L2 and Elastic. The other one is regularization strength C. The C is bigger, the more penalty to large coefficient. All this parameter will be chose by validation set. C is sample evenly for 36 points in logarithm field in [-5, 10] which can help to have wider searching range for C. Parameter θ will be got by training.
- Logistic regression is a kind of generalized linear model, we can use the number of independent parameters in the model to measure its complexity. In TF-IDF pipeline, the model complexity is 3192, the feature number after PCA. In Word2Vec pipeline, the model complexity is 50.

Linear Regression with L2 penalty (Ridge Regression)

- Linear regression has model as below:

$$\hat{y} = w^T\boldsymbol{x} + n, n \sim N(0, \sigma^2)$$

$$\hat{y}_{class} = f_{quantize}(\hat{y})$$

$f\ is\ a\ quantize\ function\ as\ below,$

$$f_{quantize}(x) = \begin{cases} 1, x \leq 1 \\ 2, 1 < x \leq 2 \\ 3, 2 < x \leq 3 \\ 4, 3 < x \leq 4 \\ 5, x > 4 \end{cases}$$

Class-weighted Loss function:

$$J = \sum_{i=1}^{N} a_{weight}\left|\left|y_i - w^T \boldsymbol{x_i}\right|\right|_2^2 + \lambda\left|\left|w\right|\right|_2^2$$

$$a_{weight} = \frac{number\ of\ data\ points\ of\ class\ l}{total\ data\ point}$$

- This model is used for baseline.
- λ is the only hyper parameter to be tuned here.
- In TF-IDF pipeline, the model complexity is 3192. In Word2Vec pipeline, the model complexity is 50.

Decision Tree Classifier

- I implement decision tree classifier by using CART algorithm in Sciket Library, which is the same algorithm in EE660. Here I use class-weighted loss for multi class classification. So, the loss function I use here is

$$\begin{aligned} f_{obj}^{k} = f_{obj}^{k-1} + [&\sum_{l=1}^{5} w_l cost\{(\boldsymbol{x_i}, l) \in R_{m_1}\} \\ &+ \sum_{l=1}^{5} w_l cost\{(\boldsymbol{x_i}, l) \in R_{m_2}\} \\ &- \sum_{l=1}^{5} w_l cost\{(\boldsymbol{x_i}, l) \in R_{m}\}] \end{aligned}$$

$where\ the\ cost\ function\ can\ be\ gini\ index\ or\ emtropy$

- Decision tree classifier is discriminative method which provide a very flexible decision boundary. For this task purpose, decision tree can be a good choice for classification. Also, this method provides the importance rank of features.
- There are 7 hyperparameters in Decision Tree classifier according to Sciket Learn Library. Here I use some prior knowledge of this task to determine some hyperparameters and the order hyperparameter tuning. In the tuning process, I will use greedy search to find out the optimal hyperparameter.

  1."criterion": It defines the cost function we use in CART. Here I will choose Gini Index cost by some intuition. Fig. 5 [9] shows the relation between class condition probability and classification cost. We can see the cost of Entropy loss is slightly larger than Gini Index, which means Entropy loss will more prefer pure region than gini index. In this task, for TF-IDF feature, I will choose gini index since there are too many features, but for Word2Vec features, I will choose entropy loss because the feature space is more reasonably small.

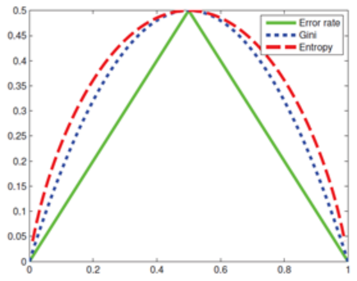


Fig. 5 cost function vs class conditional probability [9]

2."splitter": I will choose "best" here so that our algorithm will choose the most important feature to split each time. This is because the task is complicated and has lots of features. Choosing this method can provide better stability.

3. parameters selected by validation set and using greedy search.

3.1. max_depth: max_depth is the most important hyperparameter in decision, because it mainly decides the ability of the tree to fit the train set. Here I tune max_depth from 3 to 183 and fixed minimum of samples in leafs to 5, minmum of samples to split to 10. And give no constraint to maximum of feature number and minimum impurity decrease.

3.2. min_samples_leaf: min_samples_leaf can be used to smooth the boundary. I use the optimal max_length gotten from 3.1 and the search for the best min_sample_leaf from 1 to 101.

3.3. min_samples_split: min_samples_split also can be used to make the boundary smoother. It will be searched from 5 to 50.

- Before getting the training result, the complexity of a decision is not easy to measure, since we can not know the depth of a tree. Using independent parameter measurement, its complexity can be $2^{depth\ of\ tree}$.

Adaboost

- The Adaboost algorithm I use in this project is Adaboost SAMME.R [10] which is a variant of Adaboost SAMME (i.e. Adaboost M1 in EE660). The difference between these two algorithms is Adaboost SAMME.R use probabilistic estimation to update sample weight and use a different version of additive model. This improvement makes Adaboost smoother and faster. The base classifier I use here is decision tree classifier.

- Adaboost algorithm have better ability to avoid overfitting compared with logistic regression and decision. Since this project has so many features, Adaboost can provide good performance.
- There are two parameters to be tune. One is the number of estimators and the other is learning rate. The number of estimators will be tuned from 1 to 1000.
- The number of independent parameters in this algorithm will be at most 2000, in which decision stumps (maximum depth of 1 tree classifier) will provide 1000 parameters and weight of decision stump will provide another 1000 parameters.

3.5. Model Selection and Comparison of Results

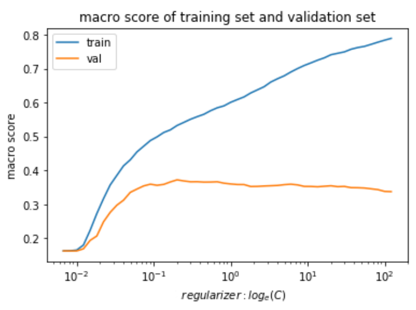
In this section, we use F1-macro score to implement model selection. The formula is show as below. In this section, I will use learning curve to compare overfitting of each model.

$$F1_{macro} = \frac{1}{5}\sum_{l=1}^{5} F1_{class_l}$$

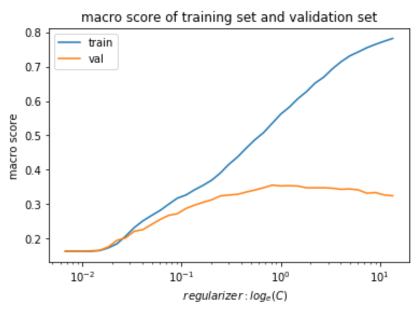
Logistic Regression:

In this section, I will plot 6 learning curves to compare the performance among 2 different features and 3 different regularizers.

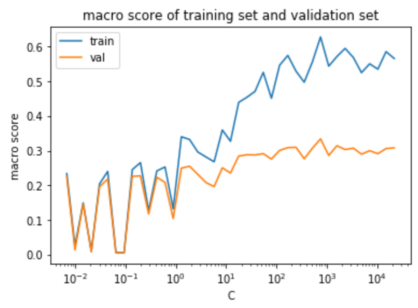
TF-IDF with PCA



L2 regularization

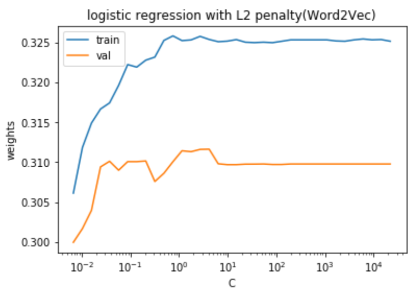


L1 regularization

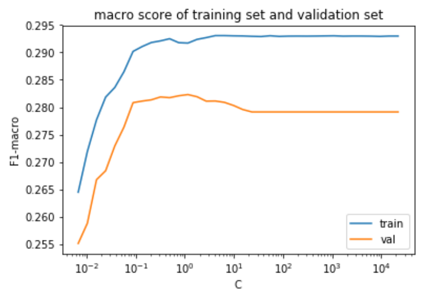


Elasticnet regularization with optimal $\alpha = 0.3$

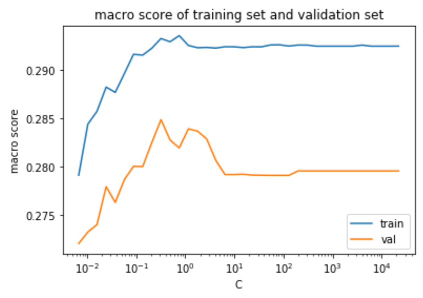
Word2Vec with SMOTE



L2 regularization



L1 regularization



Elasticnet regularization with optimal $\alpha = 0.3$

Fig. 6 Logistic regression learning curves

The $F1_{macro}$ of optimal model with these experiment settings are as blow:

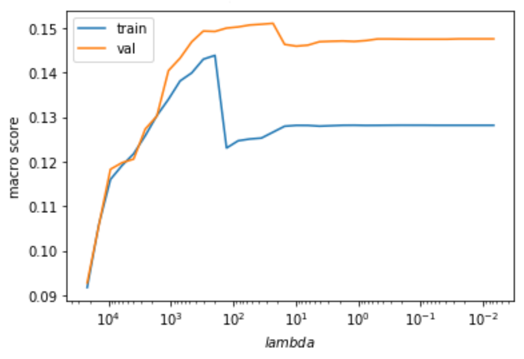
	L2-Norm	L1-Norm	ElasticNet
TF-IDF with PCA (3192 features)	**0.372**	0.363	0.312
Word2Vec with SMOTE (50 features)	0.312	0.282	0.285

Chart. 4 $F1_{macro}$ for logistic regression on Val set

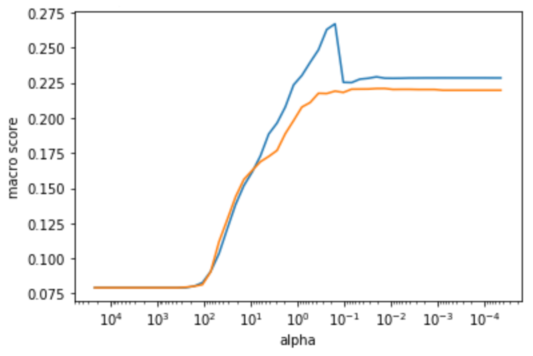
Under $F1_{macro}$, TF-IDF with PCA feature and using L2 Norm regularizer has best performance in Logistic Regression. Under each kind of feature the experiment results are consistent with expectation. Firstly, since TF-IDF with PCA is a compact feature comparing with original TF-IDF, that means each feature in TF-IDF with PCA provides information for classification. So we can see TF-IDF with L1 regularizer performs slightly worse than L2-Norm. I think the reason why TF-IDF with ElasticNet performs much worse than the other two is because ratio α is not fine tuned enough. In Word2Vec case, L2 regularizer has the best performance and ElasticNet performs slightly better than L1 regularizer. General speaking, Word2Vec feature, learned by unsupervised learning, performs worst than TF-IDF with PCA. This may be due to the training set is not big enough to learn a good embedding for each word. We can also see that TF-IDF with PCA is easier to get overfitting because of its large feature amount.

Ridge regression:

Learning curves are shown as below.



Word2Vec with SMOTE



TF-IDF with PCA

Fig. 7 Ridge regression learning curves

	Ridge Regression
TF-IDF with PCA (validation set)	0.221
Word2Vec with SMOTE (validation set)	0.151
TF-IDF with PCA (test set)	**0.219**

Chart. 8 $F1_{macro}$ for Ridge regression

The ridge regression is used as a baseline. We choose TF-IDF with PCA and use ridge regression as the baseline model.

Decision tree:

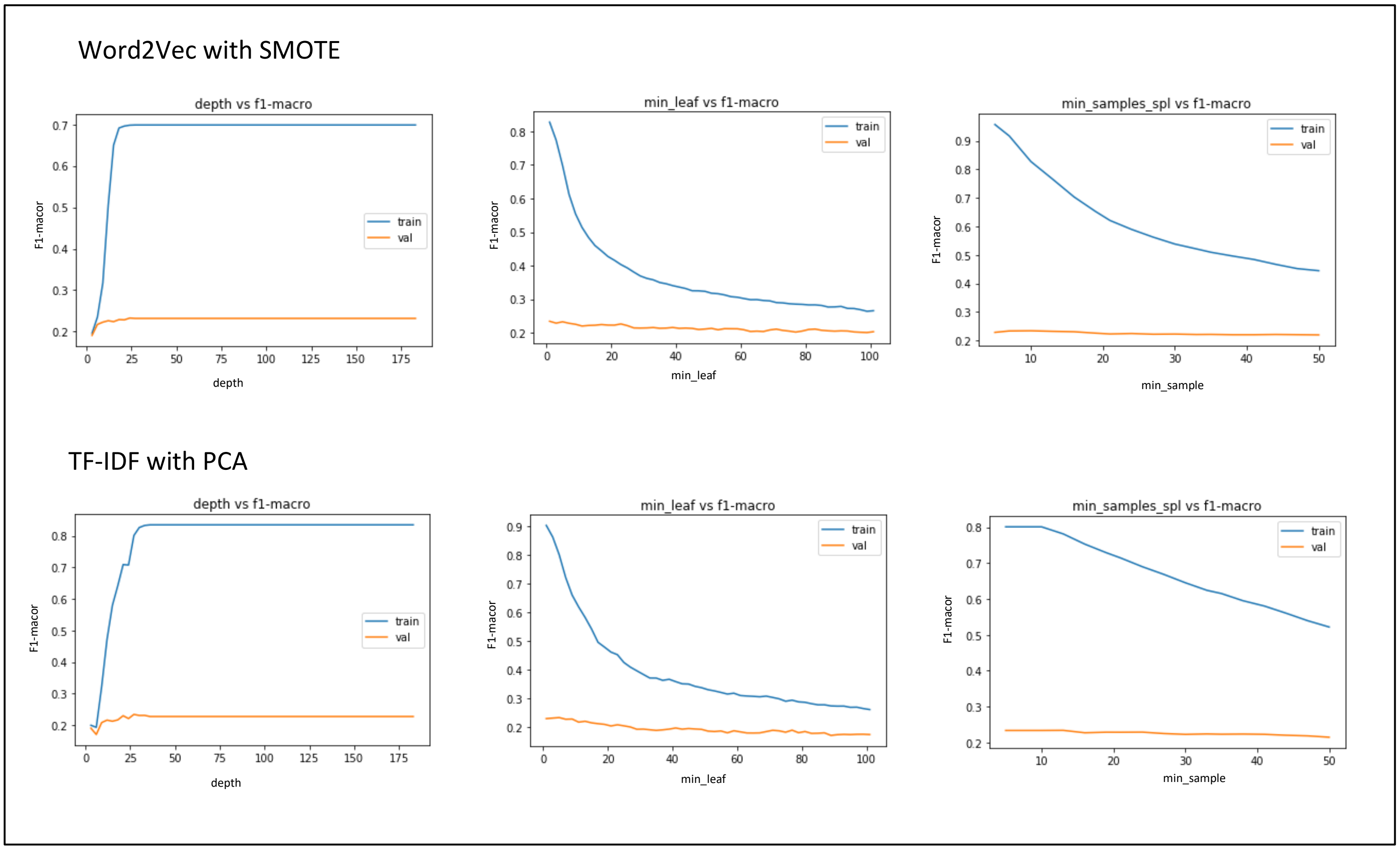


Fig. 8 learning curve for decision tree classifier

	depth		min_leaf		min_samples in each split	
	optimal parameter	F1-macro for optimal model	optimal parameter	F1-macro for optimal model	optimal parameter	F1-macro for optimal model
TF-IDF with PCA	27	0.2337	5	0.2337	13	0.2340
Word2Vec with SMOTE	24	0.2328	1	0.2343	10	**0.2343**

Chart. 9 $F1_{macro}$ for Decision Tree Classifier

In 3.4 section, we talk about the greedy training procedure of Decision Tree Classifier. Chart 9 shows the training results of each training step. Fig. 8 shows that Decision Tree are easy to get overfitting. When tree depth is above 25, F1-macro score goes to about 0.7 under both features, but the F1-macro score in validation set get stuck in 0.23 even though the model complexity goes much higher. Then I choose other two features, i.e. minimum samples in a leaf and the minimum number of samples in a node, to constrain the model complexity. But it is shameful that these constraints cannot give substantial improvement on validation F1-macro score. One possible explanation may be the greedy search. Because we greedily choose the first model with biggest validation score in depth step, the model here may be still too simple. So when we apply more constraints in the next two steps, that is reducing the variance, the score on validation set cannot get improved, since the bias of model is still too large.

Adaboost

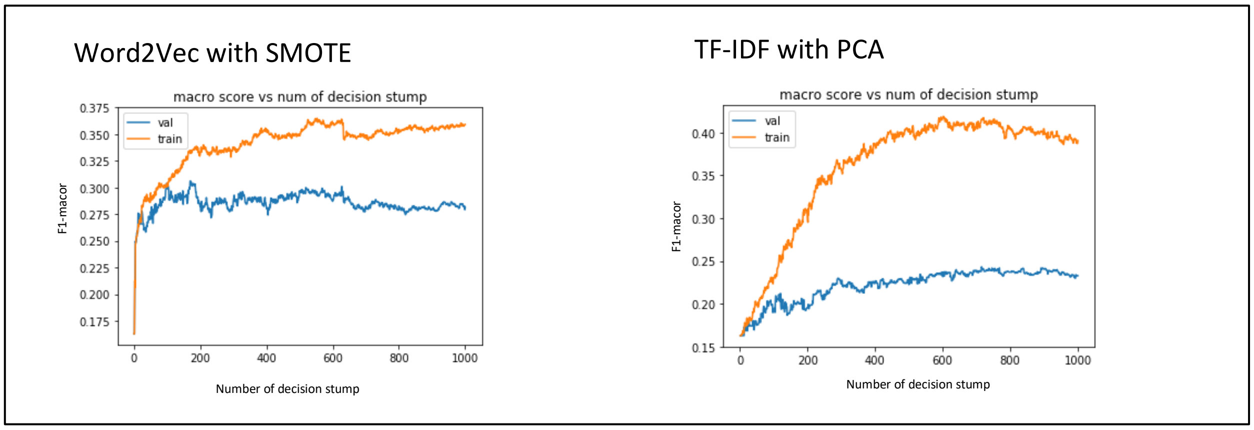


Fig. 9 learning curve for Adaboost Classifier

	number of decision stump	
	optimal parameter	F1-macro for optimal model
TF-IDF with PCA	715	0.243
Word2Vec with SMOTE	170	0.306

Chart. 10 $F1_{macro}$ for Adaboost

For Word2Vec features, the experiment result is consistent with our expectation. The Adaboost with decision tree stumps performs 30.6% better than a single finetuned decision tree. However, as for TF-IDF with PCA features, Adaboost just performs 3.8% better than the a single finetuned decision tree. One reason may be TF-IDF with PCA features still have too much variables. Even with Adaboost, the variance of the model is still too high, so we don’t get much improvement on the F1-macro score.

4. Final Results and Interpretation

Final system is Logistic Regression model using TF-IDF feature with PCA. The performance of the model is giving as below.

Optimal model: Logistic Regression model using TF-IDF feature with PCA	
Hyper-parameter: C=0.20189651799465547, penalty: L2 regularizer	
Data set: test set, size: 6926	
F1-macro score on test set	0.366
confusion matrix on test set	[[36 14 10 6 16] [18 13 16 10 24] [28 33 79 58 102] [38 45 123 411 1091] [37 37 84 348 4249]]
F1 score of each class	1:0.301, 2:0.117, 3:0.258, 4:0.323, 5:0.830
accuracy on test set	0.691

Chart. 10 Optimal model performance

Baseline model: Ridge regression using TF-IDF feature with PCA	
Hyper-parameter: $\frac{1}{\lambda} = 0.0138$	
Data set: test set, size: 6926	
F1-macro score on test set	0.219
confusion matrix on test set	[[2 19 38 21 2] [1 10 34 35 1] [1 18 112 164 5] [1 15 229 1357 106] [0 3 172 3795 785]]
F1 score of each class	1:0.046, 2:0.137, 3:0.253, 4:0.383, 5:0.277
accuracy on test set	0.327

Chart. 10 Baseline model performance

In this project, our concern is about a more balanced and better performance on classes, i.e. F1-macro score. As for this score it is not easy to get a generation bound. So, I choose the accuracy score to get a generation bound.

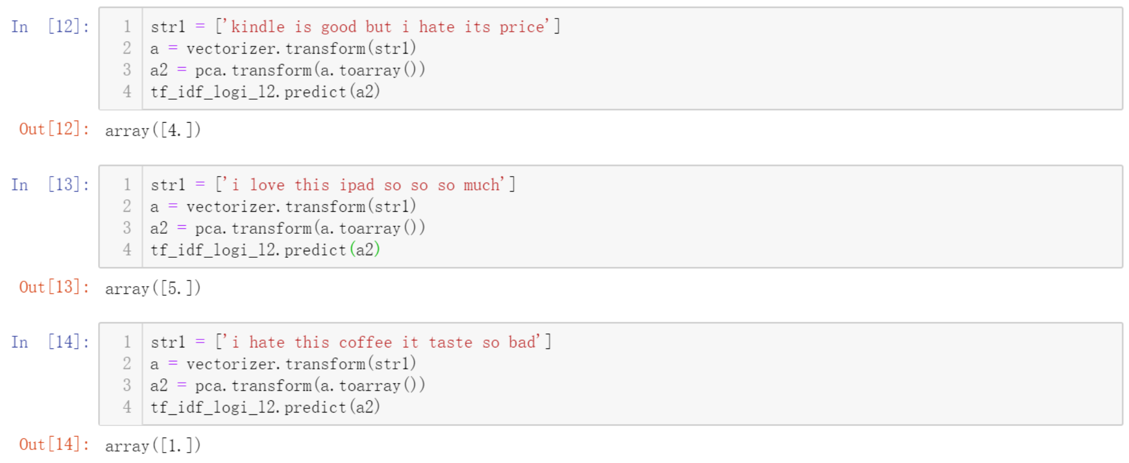
The generalization bound is given as blow,

$choosing\ \delta = 0.05,$

$$E_{out}(h_{opt}) \leq (1 - Acc(h_{opt})) + \sqrt{\frac{1}{2 \times N_{test}} \ln\left(\frac{2 \times 1}{\delta}\right)}$$

$$E_{out}(h_{opt}) \leq (1 - 0.691) + \sqrt{\frac{1}{2 \times 6926} \ln\left(\frac{2 \times 1}{0.05}\right)} = 0.325$$

Fig. 10 is a toy demo showing the performance of the model:



```
In [12]: str1 = ['kindle is good but i hate its price']
         a = vectorizer.transform(str1)
         a2 = pca.transform(a.toarray())
         tf_idf_logi_l2.predict(a2)

Out[12]: array([4.])

In [13]: str1 = ['i love this ipad so so so much']
         a = vectorizer.transform(str1)
         a2 = pca.transform(a.toarray())
         tf_idf_logi_l2.predict(a2)

Out[13]: array([5.])

In [14]: str1 = ['i hate this coffee it taste so bad']
         a = vectorizer.transform(str1)
         a2 = pca.transform(a.toarray())
         tf_idf_logi_l2.predict(a2)

Out[14]: array([1.])
```

Fig. 10 Optimal model demo

Performance Analysis:

The optimal model shows a great performance on the review with 5 stars, and the performance on reviews with 4 stars 1 stars is similar and on the second and third place. We can see that the system performs comparatively bad on reviews with 2 stars and 3 stars. And from the confusion matrix, we can see that reviews with 2 stars are classified to all five classes with similar ratio, which means the model encounter with great ambiguity on this class. So I further explore the original data in the data set, the following is two example of the reviews with rating 2.

i have always gone to buy a kindle because i like the book and th game available but i found them very unpredictable and unreliable the lor burn futon screen then the screen start jumping a a result i have purchased over the year not sure if want to go through that for the th time

we purchased this for my year old granddaughter so that she would have something to play game and read book without having to use her mom is device she took to it immediately it connects easily to a ghz network but doe not support connectivity to a search and downloads did appear a bit slow but did not do testing to see if it wa the fire or the network then again it wa running update the camera take great picture even if a child is a bit shakey sound quality is very good a is touch sensitivity

As the example shows, the reviews with rating 2 are usually very long and have lots of turns in their meanings. Even for our human being, it is not easy to predict a exactly 2 star by looking at the review.

Unexpected Result:

The most unexpected result is Word2Vec feature fails to win in the project. My first explanation is that the training data is not big enough for Word2Vec model to train a good semantic feature. The second explanation is about the average operation. Since we just take an average on the word vector to obtain a vector for a whole sentence, many good features in the word may be diluted and loss a lot of the information.

Further experiment:

In further experiment, I think we need to construct a better noise cleaning pipeline and reconstruct the target of the project. I think the five level review rating is to much to reflect the emotion in the review text, because it is a usual case that we leave a really unhappy review but finally give a 2 star rating. So it will more reasonable to combine the reviews with 2 stars and reviews with 1 stars and the same to those with 3 stars and 4 stars. As for improvement, I think we can try some deep learning methods, like RNN, with Word2Vec features so that we can better utilize all the information of a word.

5. Summary and conclusions

In this project, I have a better understanding on machine learning procedure and have a dozen of handful practices Logistic Regression, Ridge Regression, Decision Tree Classifier and Adaboost algorithm. These experiments help me to understand how to avoid overfitting, how to choose hyperparameter and how to find a suitable model for a specific problem. Finally, I am very happy to build a system to predict sentiment Amazon Review.

6. References

Cite the sources of your information that came from elsewhere. This includes other related works you compare with (if any), and sources of descriptions of systems or methods that you included in your report.

[1] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural

information processing systems (pp. 3111-3119).

[2] Pennington, J., Socher, R. and Manning, C., 2014, October. Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543).

[3] Moraes, R., Valiati, J.F. and Neto, W.P.G., 2013. Document-level sentiment classification: An empirical comparison between SVM and ANN. Expert Systems with Applications, 40(2), pp.621-633

[4] " Consumer Reviews of Amazon Products," 19 May 2019. [Online]. Available: https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products.

[5] " Datafiniti's Product Database," 19 May 2019. [Online]. Available: https://datafiniti.co/products/product-data/.

[6] " Sentiment Analysis using Python," 2017. [Online]. Available: https://www.kaggle.com/sasikala11/sentiment-analysis-using-python.

[7] " sklearn.feature_extraction.text.TfidfVectorizer," [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[8] " Word2Vec Model," [Online]. Available: https://radimrehurek.com/gensim/auto_examples/tutorials/run_word2vec.html#sphx-glr-auto-examples-tutorials-run-word2vec-py.

[9] Murphy, K.P., 2012. Machine learning: a probabilistic perspective. MIT press.

[10] Hastie, T., Rosset, S., Zhu, J. and Zou, H., 2009. Multi-class adaboost. Statistics and its Interface, 2(3), pp.349-360.

7. Appendix

Appendix I.

- **Please download pca.pkl and place it in input/model folder**. Run main file and test set score will be print.
- If you want to check all the validation set result and retrain the model, please run main_extend.py
- You can train and represent all the result by running jupyter-notebook tf-idf.ipynb, tf-idf1.ipynb, word2vec.ipynb and word2vec1.ipynb. Preprocessing pipeline can be reimplemented by running preprocess.ipynb.

- Embedding folder stores Word2Vec model for this project and input folder stores the dataset that are well separated. The folder input/model stores the optimal model using TF-IDF feature of each algorithm under validation set. The folder input/model_w2v stores the optimal model using Word2Vec feature of each algorithm under validation set.
- All the training is being done on Ubuntu 18.04 with sciket learn 0.21.3 and numpy 1.16.2. The hardware platform is AMD R5 2600 with 16GB RAM.

Folder organization: The file in red color need to be downloaded.

Please remember to download **`pca.pkl`** `so that main.py can be ran.`

You can download from:

https://1drv.ms/u/s!AvsmdpbiZEtGgb5yWQ5i_g98iIaxMg?e=KRKfia

```
├── main.py
├── input
│   ├── test_rate.csv
│   ├── test_review.csv
│   ├── test_review.txt
│   ├── train.csv
│   ├── train.txt
│   ├── val.csv
│   ├── val.txt
│   ├── y_train.csv
│   ├── y_val.csv
│   ├── model
│   │      ├── adaboost1.pkl
│   │      ├── decision_tree.pkl
│   │      ├── logistic_elsticnet.pkl
│   │      ├── logistic_L1.pkl
│   │      ├── logistic_L2.pkl
│   │      ├── pca.pkl
│   │      ├── tfidf.pkl
│   │      └── tf-idf_ridge.pkl
│   └── model_w2v
│          ├── w2v_adaboost.pkl
│          ├── decision_tree_w2v.pkl
│          ├── logistic_elsticnet.pkl
│          ├── logistic_L1.pkl
│          ├── logistic_L2.pkl
│          └── w2v_ridge.pkl
├── embedding
│   └── word2vec.model
├── w2v_train
│   ├── preprocess.ipynb
│   ├── tfidf.ipynb
│   ├── tfidf1.ipynb
│   ├── train_w2v.ipynb
│   ├── word2vec.ipynb
│   ├── word2vec1.ipynb
```

```
|     ├── preprocess.py
|     ├── tfidf.py
|     ├── tfidf1.py
|     ├── train_w2v.py
|     ├── word2vec.py
|     ├── word2vec1.py
```