Supervised Text Classification for Fine-grained Sentiment Analysis of Online Reviews

DS5220 Fall 2018, Group 1

Abstract—In response to an AI Competition which calls participants to predict sentiment labels for a massive online reviews dataset, we tested various supervised text classification methods for fine-grained sentiment analysis, evaluating their prediction performance, training speed and ease-of-use.

1 INTRODUCTION

Propelled by the prevalence of e-commerce and online review sites, online customer reviews have become an important source of information for customers decision making [1]. For a specific type of service or product, the reviews often focus on a set of relevant aspects. For example, restaurant reviews normally would only mention food taste, service, and dining environment. Being able to dissect reviews based on these aspects helps businesses understand customer experience faster and in greater details, hence deriving more informed marketing strategies and product/service improvement plans.

Using a massive dataset of restaurant reviews, we looked into ways to apply supervised classification models in the context of fine-grained sentiment analysis. We also experimented various feature extraction methods for text data and tuned the hyperparameters of best-performing models to find the balance between training speed and prediction accuracy. Our final model with TF-IDF vectorization and Linear SVC was able to achieve an average macro-F1 score of 0.52.

1.1 Dataset

We obtained the dataset from AI Challenger 2018 [2], a global programming competition for artificial intelligence. The original dataset has 120,000 restaurant reviews in Chinese,[1] scraped from Dianping.com (a Yelp-like review site for local businesses). They are manually labeled for 20 fine-grained sentiment aspects under six categories, including location, service, price, environment, dish and others (see Appendix I). Each aspect is labeled with `-2` (not mentioned), `-1` (negative), `0` (neutral) and `1` (positive).

Since most reviewers do not write extensively for all aspects of their experience, we have some very imbalanced labels in some of the aspects.

2 METHODS

2.1 Feature Extraction

To transform text corpus into machine understandable numeric features, we utilized various vectorization techniques, including a couple of bag-of-words models, term count, TF-IDF, hashing vectorization, as well as Word2Vec representations.

The Term Count model simply counts the word frequency in each document and build a $n \times p$ matrix where n is the sample size, and p is the vocabulary size. TF-IDF (Term frequencyInverse document frequency) assign weights to each term based on their document frequency and their frequency in current document, basically a term appears less frequent in other document, but very frequent in current document will receive a higher weight.

As the vocabulary size in a typical language is very big, we often needs to reduce the dimensionality of bag-of-words features in order to make the training speed manageable. Bag-of-words representation in combination with dimension reduction is what is called *Latent Semantic Analysis (LSA)*. LSA assumes that words with close meanings occur in similar pieces of text. A typical dimension reduction method used for LSA is Singular Value Decomposition, as it works well with large dataset and sparse matrices.

We also tested a few contextual text features, such as Latent Dirichlet Allocation and Word2Vec. Latent Dirichlet Allocation works well for topic extraction, but the topics extracted were later proved to be not a very good candidate for our multi-label classification task.

Word2Vec generates a vector of fixed dimension for each word, based on the context where we have seen for that word. In order to utilize word vectors for a review document of many words, we simply take the averages of the word vectors in every document. As reported by Perone et al [3], this bag-of-words approach actually outperforms more sophisticated models in many linguistic probing tasks.

2.2 Model Evaluation

Each review in our dataset has 4-class multinomial labels for each of the 20 sentiment aspects, so this is a multi-output multi-class text classification problem. As our data is imbalanced in some of the sentiment aspects, we use F_1 score to evaluate classification performance.

Although in the practical settings, we might want to add different weights to some class labels or sentiment aspects, as incorrectly predicting for some aspect or class might not be as serious a problem as some others, the final performance evaluation we chose is to take the unweighted averages of the F1 scores of all 20 multinomial classifiers. That is

$$F_{1\_score\_mean} = \sum_{n=1}^{20} \frac{F_{1\_score(i)}}{20}$$

1. Not including another 120,000 testing reviews of which the text is available, but labels were held out by the organizers for competition evaluation.

$$F_{1\_score(i)} = \left(\sum_{j=1}^{4} 2 \times \frac{precision_{(j)}^{(i)} \times recall_{(j)}^{(i)}}{precision_{(j)}^{(i)} + recall_{(j)}^{(i)}} \right) /4$$

This forces our model to perform well in all scenarios, and is also what is used by the AI Challenger competition.

2.3 Basic Workflow

To make things easier to manage, we simply consider each sentiment aspect an independent classification problem. In the model selection process, we will use the same base classifier to train the same features obtained from certain feature extraction process. Then we predict the labels for each sentiment aspect one by one.

We first tested different text features on a set of popular classifiers, using a bigger sample size, but without cross-validation.

Then after identifying determining factors in feature extraction process and the base classifiers with the most potentials, we then tuned the hyperparameters in these two process to search for the model with the best performance.

As hyperpameter searching takes a lot of time, we have chosen to use smaller sample size in this stage to speed up the model selection process.

2.4 Base Classifiers

Initially, we tried a wide range of classifiers, including Dummy classifier, Nave Bayes Linear Discriminant Analysis, Quadratic Discriminant Analysis, Logistic Regression, Linear SVC and Ridge Classifier.

Dummy classifier is just a baseline model where we always return the majority class labels based on stratified class frequency. This is used as a simple baseline.

Naive Bayes classifier is a conditional probability model using Bayes' theorem. We tried two kind of Naive Bayes classifier in our project. One is Multinomial Naive Bayes and the other is Complement Naive Bayes. Multinomial Naive Bayes is popular to do document classification where binary term occurrence features are used rather than term frequencies and Complement Naive Bayes was designed to improve Multinomial Naive Bayes classifier by correcting severe assumptions problem. Complement Naive Bayes is widely used for unbalanced data classification.

Linear Discriminant Analysis (LDA) assumes the conditional probability density function are normally distributed then use the Bayes optimal solution to predict points as being from the second class if the log of the likelihood ratios is bigger than some threshold. If there is no normally distributed assumption, it goes to Quadratic discriminant analysis(QDA).

Both Logistic Regression and Linear SVC were solved with Stochastic Gradient Descent, as proved in our experiment, with appropriate hyperparameters SGD trains much faster on large dataset with almost identical performance than non stochastic version the corresponding classifier.

3 RESULTS

3.1 Preliminary results

There are 9 features and 9 classifiers included in the preliminary analysis. The 9 features are word counter with different vocabulary size (`min_df = 0.02, 0.01`), word counter combined with tf-idf with different vocabulary size (`min_df = 0.02, 0.01`), word counter combined with tf-idf and SVD(`n_components = 500, 1000`) with different vocabulary size (`min_df = 0.02, 0.01`) and word2vec. The 9 classifiers are multinomial Nave Bayes, complement Nave Bayes, dummy, dummy stratified, LDA, QDA, logistic regression (stochastic gradient descent), SVC (linear kernel, stochastic gradient descent), Ridge. 10,000 samples are randomly selected to train the model.The performance reported are achieved on development set.

Generally, the overall performance in the Table 1 indicates that LDA, SVC and logistic regression are the top three models with the best performance. There are some NAs in the table which is due to the lack of ability of complement Nave Bayes to handle negative input values.

The Table 2 presents the average training speed for each model. The speed of complement Nave Bayes classifier is closed to that of dummy classifier model, which implies the potential of the complement Nave Bayes to act as classifier in the baseline model. Furthermore, whenever included the SVD, the model tends to take more time, which demonstrated that SVD is time consuming step. However, there is two exceptions in the table. The model with SVC (stochastic gradient descent) and logistic regression (stochastic gradient descent) are less influenced by the SVD step as showed in the table. XX. Additionally, The LDA and ridge classifier are extremely slow compared with other classifier options.

3.2 Fine tuning results

3.2.1 Features extraction

In order to find the balance between information and training speed for our model, we tuned features of the counter vectorizer and singular value decomposition (SVD) before we get to the classifier step. Using the Google translated English data set with 1000 random samples, we conducted 5-fold cross validation grid search. To compare the performance of each model, we adopted LDA as the classifier.

First, we set the number of SVD components to 200 and tuned the minimal document frequency (min_df) and n-grams range. From the Fig. 1, we can see that as min_df increase, the F1 scores of the model keep rising and achieve the highest value when min_df in the range 0.05-0.08 for all n-grams values. While, the behavior of n-grams range are similar among different values. From this grid search, we find the combination of min_df equals to 0.05 and uni-gram produce the best performance. However, since we only use 1000 samples, once the sample size become larger, the combination may change.

Second, we used the uni-gram which gave the best results from the previous grid search for the counter vectorizer and tuned the number of components for the SVD. The Fig.2 shows that all numbers of components larger than 100 produced quite good results. The model with min_df equals to 0.05 and 250 components for SVD outperformed others. The value of min_df chosen from this grid search is same with that from the first tuning, which confirmed that 0.05 is the best min_df value we could have under this model condition.

To search the best number of SVD components, we also plot the F1 scores with the number of components, which can be found in Fig. 3. There is an obvious plateau when the number of components is larger than 200, which is consistent with our

TABLE 1: Preliminary Model Performance

	count	count_sv	tfidf	tfidf_sv	lsa_500	lsa_500_sv	lsa_1k	lsa_1k_sv	word2vec
LDA	NaN	NaN	0.513	0.506	0.479	0.474	0.502	0.501	0.446
SGD_SVC	0.445	0.429	0.513	0.503	0.463	0.455	0.489	0.477	0.378
SDG_Logistic	0.439	0.429	0.505	0.499	0.462	0.471	0.479	0.480	0.352
ComplementNB	0.417	0.403	0.424	0.410	NaN	NaN	NaN	NaN	NaN
Ridge	NaN	NaN	0.413	0.403	0.384	0.380	0.396	0.392	0.358
DummyStratified	0.251	0.246	0.248	0.245	NaN	NaN	NaN	NaN	NaN
DummyMostFrequent	0.199	0.199	0.199	0.199	NaN	NaN	NaN	NaN	NaN

TABLE 2: Training Speed of Preliminary Models

	count	count_sv	tfidf	tfidf_sv	lsa_500	lsa_500_sv	lsa_1k	lsa_1k_sv	word2vec
DummyMostFrequent	0.135	0.139	0.155	0.143	NaN	NaN	NaN	NaN	NaN
DummyStratified	0.200	0.170	0.176	0.205	NaN	NaN	NaN	NaN	NaN
ComplementNB	1.157	0.981	1.098	1.024	NaN	NaN	NaN	NaN	NaN
SGD_SVC	8.642	7.665	6.275	6.318	13.526	13.301	25.440	25.194	10.157
SGD_Logistic	8.628	7.395	7.003	5.827	15.801	15.361	29.475	27.155	52.316
LDA	NaN	NaN	NaN	NaN	58.567	54.765	148.312	143.328	30.537
Ridge	NaN	NaN	NaN	NaN	69.216	65.362	166.688	174.574	39.619

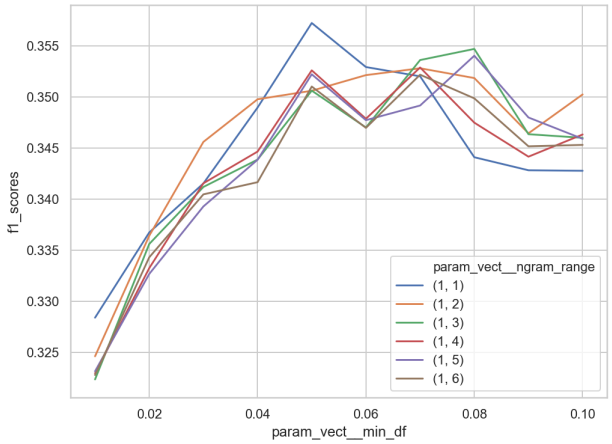

Fig. 1: The model performance with different minimal document frequency and n-grams range

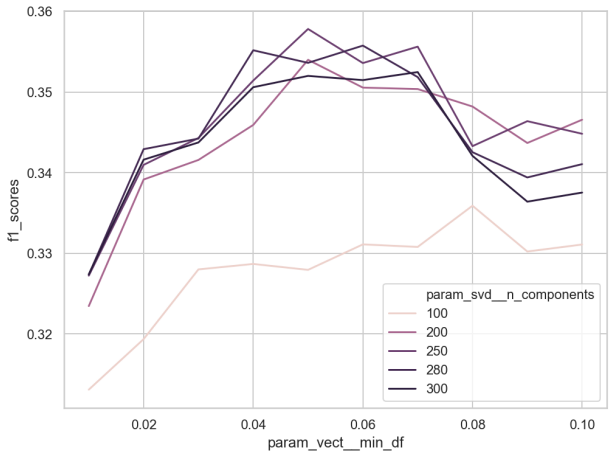

Fig. 2: The model performance with different minimal document frequency and number of SVD components

previous grid search. Meanwhile, the graph also illustrates the different trends between training and test set when the number of components exceed its optimal value.

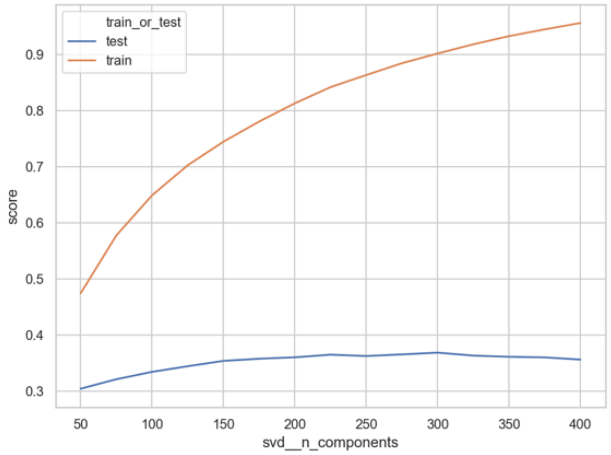

Fig. 3: The relationship between model performance and number of SVD components for training and test set

3.2.2 *Base classifiers*

Based on the best feature parameters we discovered previously (uni-gram, min_df = 0.05, number of components = 250), we tuned the parameters for logistic regression, SVM and neural network, which are three best models from the preliminary model selection. We did not choose the LDA here because its parameter is related to the features that have been tuned in the feature extraction section. To compare the performance of each model, we used 1000 random samples from Google translated data set as the feature extraction.

The model performance with different kind of penalty and weighted methods can be found in Fig. Fig. 4. The mean value of F1 scores with L1 penalty is about 0.345 which is higher than the mean value of F1 scores with L2 penalty which is about 0.338. F1 scores achieved from model with weights is higher than that without weights for both L1 and L2 penalty. It is clear that

logistic model with weighted classes and L1 penalty gives better performance.

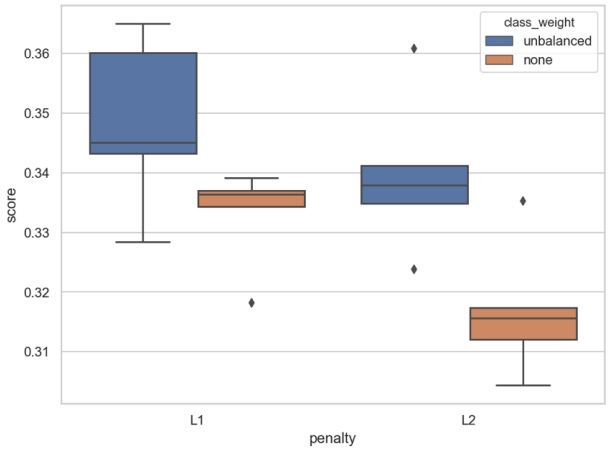

Fig. 4: The model performance with and without weighted classes and L1/L2 penalty

For the SVM (Fig. 5), we explored the performance of models with different values of penalty and different kernels. As the value of penalty increase, the F1 scores for all models except the one with poly kernel increase at the beginning, and reach the peaks and become stable successively. It is apparent that models with large penalty tends to have better performance. Moreover, the radial basis function kernel turns out to be the best choice when penalty is larger than 60.

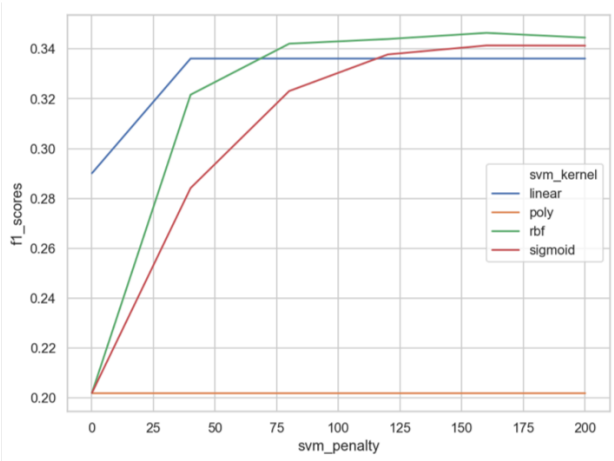

Fig. 5: The relationship between model performance and penalty value with different kind kernels

The Fig. 6 shows that logistic activation function gives the lowest F1 scores, which is around 0.23. Tanh and Relu activation functions give higher F1 scores around 0.27. However, it is still quite low compared to other models, which might result from the small sample size being used. As the size of hidden layer increase, F1 scores of the model slightly increase and tend to reach a plateau. For large sample size, the size of hidden layer basically has no influence on the F1 scores.

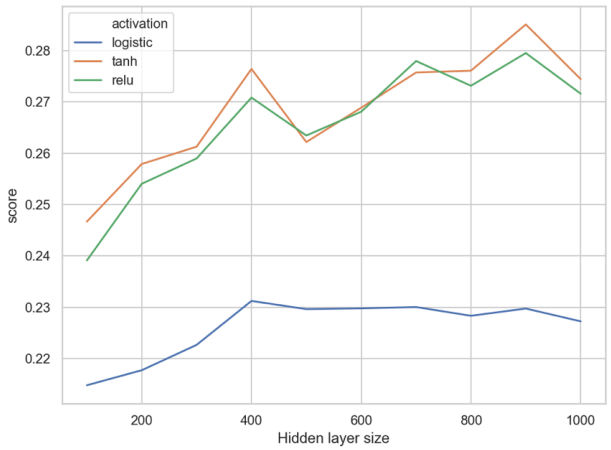

Fig. 6: The relationship between model performance and hidden layer size with different activation functions

4 DISCUSSION

4.1 Treatment of unbalanced data

Most of the aspects in the data set are highly unbalanced, which should be treat carefully. During the modeling process, the baseline model adopted the complement Nave Bayes which has embed method to deal with the unbalanced data and produced a quite good F1 scores considering its training speed. In addition, the different results from weighted or non-weighted logistic regression also demonstrated the importance of extra approaches to handle the unbalanced data. In fact, there remains many methods dealing with unbalanced data, such as over sampling and down sampling as well as taking average of multiple times of either those two sampling flavor. Since these sampling methods need complex feature engineering as each aspect has unique data distribution, we did not included them in our modeling process.

4.2 speed/performance trade off

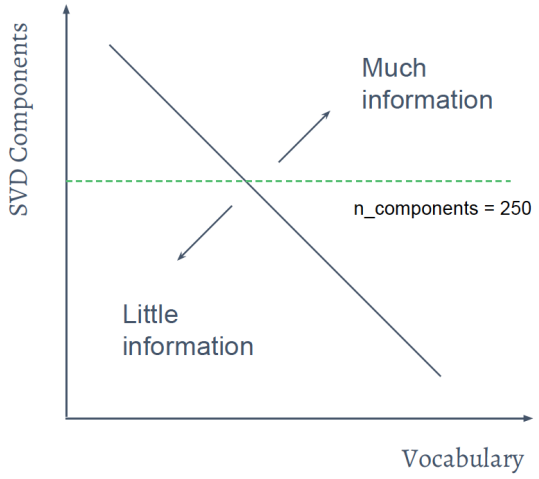

Fig. 7: information and training speed trade off

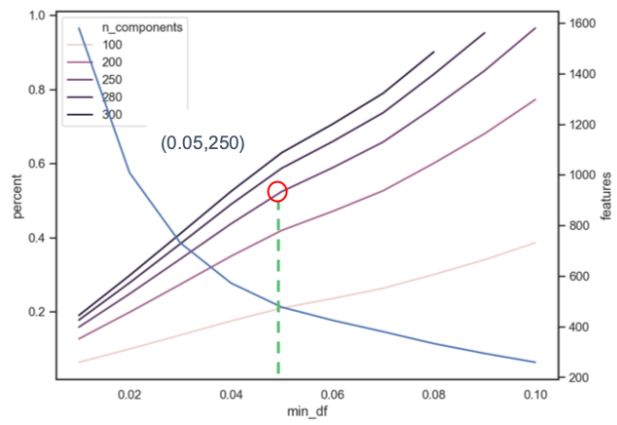

Fig. 8: The relationship between minimal document frequency and vocabulary size and information completeness

4.3 signal/noise trade off

4.4 future work

REFERENCES

[1] Zhang, Kem ZK, et al. "Examining the influence of online reviews on consumers' decision-making: A heuristicsystematic model." Decision Support Systems 67 (2014): 78-89.

[2] Challenger AI, https://challenger.ai/?lan=en

[3] Perone, Christian S., Roberto Silveira, and Thomas S. Paula. "Evaluation of sentence embeddings in downstream and linguistic probing tasks." arXiv preprint arXiv:1806.06259 (2018).

APPENDIX

Term Definition

SVD: Singular-Value Decomposition, a matrix decomposition method for reducing a matrix to its constituent parts in order to make certain subsequent matrix calculations simpler. The calculation is based on

$$A = USV,$$

where:

- A is an $m \times n$ matrix
- U is an $m \times n$ orthogonal matrix
- S is an $n \times n$ diagonal matrix
- V is an $n \times n$ orthogonal matrix

References

TABLE 3: All 20 Sentiment Aspects

location	traffic convenience distance from business district easy to find
service	wait time waiters attitude parking convenience serving speed
price	price level cost-effective discount
environment	decoration noise space cleanness
dish	portion taste recommendation look
others	overall experience willing to consume again

STATEMENT OF CONTRIBUTIONS

- **Jianchao Yang** - Coded the framework of the training process. Built the visualization.
- **Zishen Li** - Fine tuning multiple models.
- **Xinyu Tang** - Fine tuning neural network.
- **Kefei Zhan** - Feature Extraction