Fine-Grained Sentiment Analysis of Restaurant Customer Reviews in Chinese Language

Suofei Feng[1] *suofeif@stanford.edu*
Eziz Durdyev *eziz@stanford.edu*

***Abstract*—Chinese language processing is a challenging topic in the well-developed area of sentiment analysis. In this project we explore Long Short-term Memory neural network with different word representations for the fine-grained, or aspect-level sentiment analysis of restaurant customer reviews in Chinese language. There are 20 aspects for classification, each representing one type of target information in the reviews. We separately train one model for each element. The accuracies, weighted F1 scores, and confusion matrices are referred to evaluate the models. The results of different models vary across the 20 aspects.**

***Index Terms*—Chinese language, NLP, LSTM**

I. Introduction

THE era of information explosion, brings an increasing demanding on the ability to extract core message from billions of records of data. Sentiment analysis, or opinion mining, is widely applied to extracting and studying subjective information in texts. By quantifying the opinions or attitudes in a large bulk of texts in a few minutes, sentiment analysis has gained popularity in various business scenarios for retrieving customer responses. In recent decades, considerable progress has been achieved in sentiment analysis of English language. However at the same time, a similar development comparable to the growth of market is not seen in Chinese language applications [1]. In our project, we propose to implement a fine-grained (aspect-level) sentiment analysis of restaurant customer reviews in Chinese language. The topic and data come from the 2018 AI Challenger competition [2].

The inputs are reviews about restaurants in Chinese language. The task is to classify each piece of review text into 4 classes (”not mentioned”[-2], ”negative”[-1], ”neutral”[0], ”positive”[1]) under 20 aspects. Each aspect (or element) represents one type of information about the business. In the scope of each model architecture, we develop and train separately one model for each of the aspects.

[1]Department of East Asian Languages and Cultures, Stanford University.

II. Related Work

A. Approaches to Sentiment Analysis

Pang and Lee [3] briefly summarize the history of sentiment analysis, and describe the related works as computational treatment of ”opinion, sentiment, and subjectivity in text” (p8). Early machine learning approaches towards sentiment classification use unigrams (single words) and n-gram as features for subjectivity detection, and SVM, MNB (Multinomial Naïve Bayes), etc. for classification [4] [5].

Recent years have seen a substantial progress in NLP tasks with neural network approaches. LSTM is popular in sequence modeling for sentiment classification because of its advantage against gradient vanishing or exploding issues in long texts. Wang et al. [6] proposed an attention-based LSTM model with aspect embedding for aspect-level sentiment classification. They experimented with restaurant customer reviews in English, and implemented a 3-class (positive, negative, neutral) classification on 5 aspects: food, price, service, ambience, anecdotes/miscellaneous. The accuracy of this model improved by 2% compared with standard LSTM model. However, considering the different natures of Chinese language, and the large number of aspects for classification (20), we decide to start with standard LSTM model for the neural network approach in this project.

B. Chinese NLP Researches

A review of sentiment analysis in Chinese language was given by Peng et al [7]. Apart from conducting sentiment analysis directly on Chinese language, there is another approach: transform the task to sentiment analysis on English language by machine translation. In our project, we conduct a mono-lingual experiment by directly extracting features from original Chinese language. This is mainly because that the style of our input texts is highly colloquial and context-specific, which might lose information in the process of machine translation. According to this article, good results were gained from a combination of neural network (word2vec) for word representation and SVM for classification.

Peng et al. also mentioned the different techniques for segmentation. As Chinese language does not have space between words, it is necessary to use segmentation tools to extract words as the basic units of semantic meaning. They summarized that Jieba had a high speed and good adaptation to different programming languages. For these reasons, we decide to use Jieba as our segmentation tool.

III. Dataset and Features

A. Dataset

We use the data sets provided by AI challenger official [2]. The training and validation data are manually labelled. They also provided test dataset without labels. In the training dataset, there are 105,000 records of reviews, with labels of 4 classes {"positive"[1], "neutral"[0], "negative"[-1], "not mentioned"[-2]} on 20 aspects/elements under 6 categories. The validation set has 14998 records of reviews. For the aim to evaluate our models by ourselves, we split the validation set into a smaller validation set (first 7500 records in the original validation set) and a test set (rest 7498 records in the validation set) with true labels. The class distributions of the 20 aspects are very similar across all three datasets. Table I shows all the elements and corresponding categories. Here is an example input text. Because of the limited space, we just put part of the review and its translation here:

"吼吼吼，萌死人的棒棒糖，中了大众点评的霸王餐，太可爱了。一直就好奇这个棒棒糖是怎么个东西，大众点评给了我这个土老冒一个见识的机会。看介绍棒棒糖是用德国糖做的，不会很甜，中间的照片是糯米的，能食用，真是太高端大气上档次了..."

Translation:
"Ha ha ha, the lollipop is soooo cute. I won the 'free meal prize' on Dazhongdianping [author's comment: similar to Yelp], this is so cute. I have been always curious about what the lollipop is like. Dazhongdianping gave me the bumpkin this opportunity to open my eyes. The introduction said it was made using German candy, not too sweet. The photo in the middle is made of glutinous rice, edible. It is really high-end..."

A glance of Google Translation:
"Hey, the lollipop of the dead man, the overlord meal of the public comment, so cute..."

TABLE I: Elements

Location	traffic
	distance from business district
	easy to find
Service	wait time
	waiter's attitude
	parking convenience
	serving speed
Price	price level
	cost-effective
	discount
Environment	decoration
	noise
	space
	cleanness
Dish	portion
	taste
	look
	recommendation
Others	overall experience
	willing to consume again

B. Feature Extraction

The main challenge in our project is preprocessing our data. Chinese language is difficult to accurately segment because of the absence of space, variant lengths of words, and high flexibility of making new words. We apply same preprocessing approaches to the training, validation, and test dataset. With reasons presented in the related work section, we use Jieba cut for segmentation. After segmentation, we gain three lists of word lists produced by segmenting the lists of sentences.

Word2Vec models are used to produce word representations for the classification task. We explore two types of Word2Vec models: model A being trained on segmented train, validation, and test text data, and pretrained model B by Tencent AI Lab [8]. The model A is trained using Gensim under Continuous Bag-of-Words mode. This model assumes that similar words share similar contexts, and thus optimizes the predictions of target word given a number of context words. The optimization objective is given here [9]:

$$minimize J = -\log P(w_c|w_{c-m}, ..., w_{c-1}, w_{c+1}, ..., w_{c+m}) \quad (1)$$

where w_c is the target word, $w_{c-m}, ..., w_{c+m}$ are context words, and m is the context size.

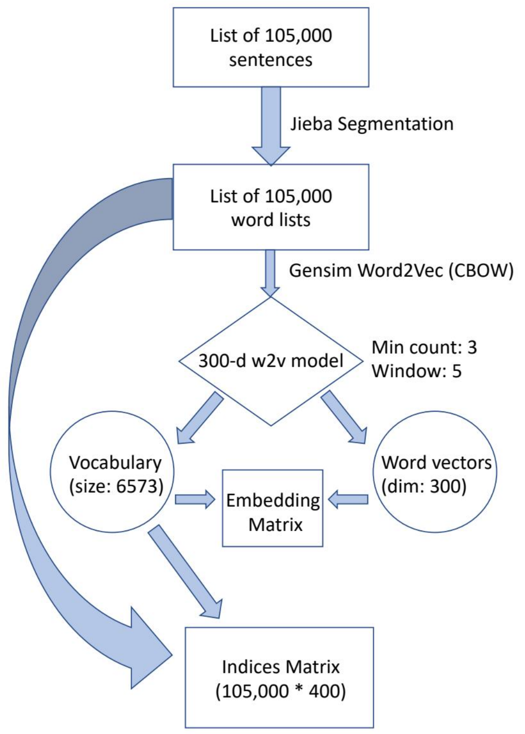

Fig. 1: Preprocessing Flowchart

We train a Word2Vec model following the instructions in [10], and produce a vocabulary (6573) and embedding matrix (6573 x 300) of the same word order with the vocabulary. Pad token and unknown-word token are added to the vocabulary as well as the embedding matrix.

The model B has a vocabulary of size 8824330, and its embedding dimension is 200. To increase the training speed, we slice out the vectors of the words present in the training, validation, and test datasets. This forms a mini embedding matrix of size (7068 x 200). As the input texts are customer reviews, which are inclined to colloquial. This means a limited sophistication in the lexicon of the input data, and further a limited improvement by training on unknown tokens. We therefore decide to also slice out the word vectors of words in test dataset for the embedding matrix.

The next step is to digitalize the input texts. We replace the words in each sentence (after segmented) with their indices in the vocabulary. All the sentences are padded or cut to a length of 400. The outputs are three matrices for train (105000, 400), val (7500, 400), and test data (7498, 400). Fig 1 shows the procedures of preprocessing training data using model A.

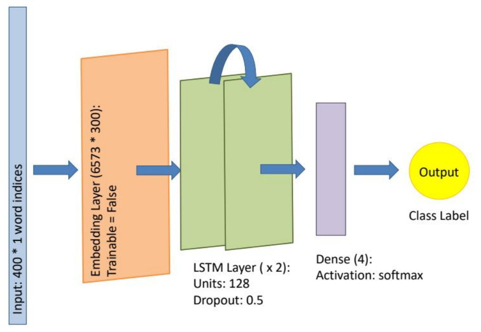
Input: 400 * 1 word indices
Embedding Layer (6573 * 300): Trainable = False
LSTM Layer (x 2): Units: 128 Dropout: 0.5
Dense (4): Activation: softmax
Output
Class Label

Fig. 2: Two-Layer LSTM Model

IV. Classification Models

A. Baseline

The baseline model is provided by AI Challenger official [11]. The feature is extracted by TF-IDF framework, whose values for representation are based on the frequency of a word in a given document. The classification model is RBF SVC. The average F1 score across the 20 elements is around 0.2.

B. LSTM

LSTM, or Long Short-term Memory network, is a type of recurrent neural network (RNN) to process sequence of information, by feeding the output of preceding neurons to subsequent neurons. Unlike traditional RNN, LSTM networks are not vulnerable to gradient explosion or vanishing when dealing with long sequences of data. This is achieved by forget gate, input gate, and output gate in each hidden unit. These gates decide how much information to let through, and therefore can connect information with a wide gap in between [12].

We firstly build a many-to-one LSTM model for each of the 20 elements. Fig 2 shows the structure of the model and the hyperparameters. The inputs are the output indices from preprocessing step. The labels are transformed to one-hot vectors, each as a (1 x 4) row vector. The embedding matrix from each of the two embedding models is used as the untrainable weights for the embedding layer. We add arbitrary class weights to address the class imbalance problem. The loss function is categorical cross-entropy:

$$L(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{4} y_{ij}\log(p_{ij}) \tag{2}$$

with n the number of examples, j the class ID, y the true label, and p the predicted probability. Accuracy and weighted F1 score are the evaluation metrics. Based on the outcomes of the previous model, we renew the model architecture and hyperparameters for further training on specific elements. The updates are discussed in the next section.

V. Training and Results

A. Training

After tuning on a subset of 500 training records and 100 validation records, we choose 0.5 for LSTM layer dropout and recurrent dropout, 128 for number of hidden units, 128 for batch size, and Adam for optimizer with learning rate of 0.001, β_1 of 0.9, β_2 of 0.999. At the beginning we tried 50 epochs for all the elements, and found most of them converge after around 14 to 15 epochs. We therefore decide to train for 20 epochs. As we have 105k records in the training dataset, batch size of 128 will make the training process comparatively fast. Adding different arbitrary class weights to different elements does not have clear improvement. The upper left plot of Fig. 3 gives the accuracies and losses of training and validation of the first element using model A for language representation. The training loss is updated for every batch, and therefore is much more unstable than validation loss.

At the beginning, we applied early stop (monitor: validation loss; patience: 3) to the LSTM using Tencent mini embedding. Soon we found many of the models stopped training after 3 or 4 epochs without reducing the loss. We then cancel the early stop for this type of models. Figures in Fig. 3 show a clear contradiction of convergence speed between A and B language models. To make this report concise and short, we will not report all the training details and statistics of all the 20 elements here.

A possible source for this different convergence process is the embedding matrix. Firstly, the vectors in the matrix of model B are from different corpus other than consumer reviews. This will prolong the process for the classification model to adapt the special theme of the input corpus. Secondly, a larger embedding size (300) of model A provides finer information of the input, and therefore makes it easier to classify. In the subsequent training and modifications, we focus on the model using representation A.

Based on the results of the models using representation A, we select 9 elements for further training. The selections are made partly due to no convergence within 20 epochs, partly due to improper class weights. We firstly train them for another 20 epochs without changing the model structures. The improvement is not clear. We then increase the number of hidden units to 256 in LSTM layers, and reduce batch size from 128 to 64 or 32. For those with batch size of 64, we increase the learning rate from 0.001 to 0.01 with a decay rate of $\frac{1}{60000}$. These modifications efficiently speed up the convergence. An example is shown in Fig. 4. For clarity, we will refer to this newly modified model as LSTM-256, the old one with representation A as LSTM-128A, the one with representation B as LSTM-128B.

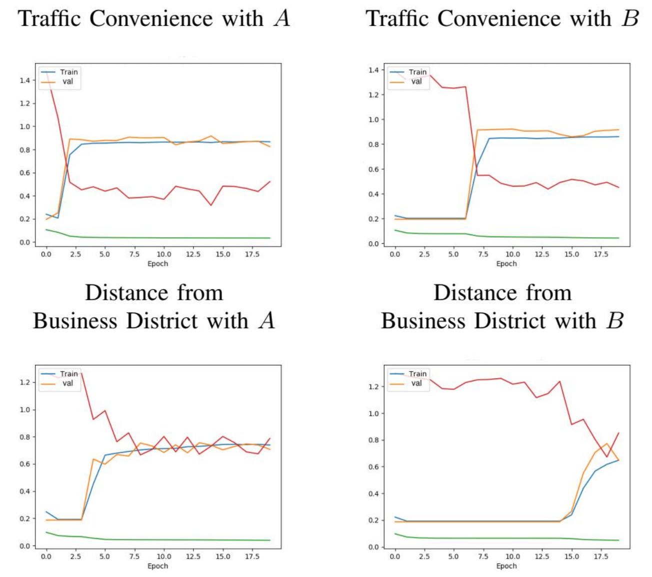

Fig. 3. These are training processes of first two elements with different word representations. In each plot, red line represents training loss; green line represents validation loss; blue line represents training accuracy; orange line represents validation accuracy.

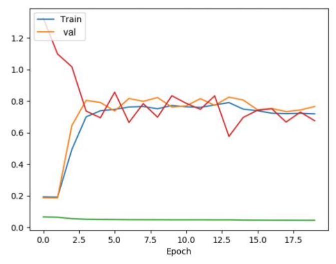

Fig. 3: Fig. 4. LSTM-256 model on Distance from Business District. Red line is training loss; green line is validation loss; orange line is training accuracy; blue line is validation accuracy.

B. Results

Our metrics include accuracy, weighted F1 score, and confusion matrix. Accuracy is a straight-forward measure of the performance of the models. However, by observation we find a serious class imbalance across all elements in the training, validation, and test data. We therefore introduce weighted F1 score for reference. Here is the function for weighted F1 score:

$$weightedF_1 = \sum_{j=1}^{4} \frac{N_j^{(T)}}{N^{(T)}} F_{1j} \tag{3}$$

where $N_j^{(T)}$ is the number of true instances of class j, $N^{(T)}$ the total number of true instances, F_{1j} the F1 score of class j. This measure takes class imbalance into account. Table II shows the average test accuracies and test weighted F1 scores across the 20 elements from the three models.

TABLE II: Test Statistics

Model	Avg. Accuracy and Variance	Avg. W. F1 Score and Variance
LSTM-128A	0.656, σ^2=0.023	0.658, σ^2=0.025
LSTM-128B	0.376, σ^2=0.054	0.288, σ^2=0.065
LSTM-256	0.670, σ^2=0.021	0.674, σ^2=0.022

The test statistics of LSTM-256 are averages of retrained elements with unretrained elements. All the three models have better performances than the baseline, though LSTM-128B is well below satisfaction. The main problem with this model, as stated in the previous section, is the speed of convergence (perhaps need more than 50 epochs for some of the elments to start converge). It is clear that the LSTM-256 performs better than LSTM-128A in terms of accuracy and weighted F1 score, with faster convergence in 20 epochs. Confusion matrices of the second element from LSTM-128A and LSTM-256 will give a tangible idea of the multi-class classification quality:

$$E2_{128} = \begin{bmatrix} 4022 & 40 & 0 & 1960 \\ 13 & 21 & 0 & 12 \\ 8 & 1 & 0 & 30 \\ 62 & 2 & 0 & 1327 \end{bmatrix}$$

$$E2_{256} = \begin{bmatrix} 4547 & 0 & 0 & 1475 \\ 31 & 0 & 0 & 15 \\ 13 & 0 & 0 & 26 \\ 136 & 0 & 0 & 1255 \end{bmatrix}$$

The first line of $E2_{128}$ means that there are 4022 of class -2 being predicted correctly, 40 of class -2 being predicted as class -1, and 1960 of class -2 being predicted as class 1. According to these two matrices, it is clear that LSTM-256 is more biased to dominating classes, while performs better on the most dominating class. But if one is more concerned with negative attitude, LSTM-128A is a better choice. Also, though neutral attitude is difficult for both models to detect, LSTM-256 is more advantageous as 'neutral' and 'not mentioned' are more similar than 'neutral' and 'positive'.

Generally the LSTM-128A model performs well when sentiment features are apparent:

"...首先说说环境还是很不错的 感觉很适合小情侣来 很温馨的感觉 喝喝下午茶感觉特别好 服务也很好哦 都很勤快 ... 中午人不多 很安静 非常喜欢这样的气氛 再说说美食 ...蛮好吃也 日式猪排饭 真的量好多 ... 玫瑰巧克力和榛果海盐拿铁真的都好好喝噢 下次再去必点 目前大众点评买单还能享受95折 真的挺划算 以后还会经常光顾的"

(Rough) Translation:
"...Environment is pretty good, suitable for couples, warm feeling. It's nice to have afternoon tea. Service is good, too... Not so many people in the noon. It's quiet. Really like the atmosphere. About food, ... pizza really good... The pork combo has a large portion...Drinks are tasty. Must try next time. Using Dazhongdianping will give you 5% discount, a good bargain. Will come often in the future."

The predictions results are shown in Table III.

TABLE III: Predictions of the Example Text.

1: -2	2: -2	3: -2	4: -2	5: 1
6: -2	7: -2	8: -2	9: -2	10: 1
11: -2	12: 1	13:-2	14:-2	15:1
16: 1	17: -2	18: -2	19: 1	20: 1

In general, class imbalance is the main obstacle to a significant improvement of performance. Class weights might be a good strategy, but the trade-off between prediction of sub-represented classes and overall accuracy requires more sophistication. On the one hand, higher precision in class weight assignments, e.g. 1/(number of class-j examples in the training data) might further improve the quality. On the other hand, data augmentation such as bootstrapping minor classes worths a try. Because of the demand on the integrity of context to judge about the sentiment, cropping is not suitable in this case.

VI. Conclusions and Future Works

Apart from the aspects mentioned in the previous section, this task can also be improved in the following aspects: 1.) collect data with higher label quality (some examples are difficult to classify even for human beings); 2.)improve the quality of language models with contextual representation, e.g. BERT; 3.) we might also benefit from applying attention mechanism for long input texts to extract key information.

VII. Contributions

Suofei is responsible for preprocessing data, constructing and training LSTM models, making the poster, and writing the report. Eziz did not enroll in this class, but still provided generous help for the completion of this project. We collaborated on CS229 course project under the same topic, and applied different methods to this task for the two courses. Both reports are submitted to the two classes.

VIII. Code

The code can be found at:

https://github.com/suofeif/CS230

This is a private repo. We have added cs230-stanford as the collaborator.

References

[1] H. Peng, E. Cambria, and A. Hussain, "A review of sentiment analysis research in chinese language," *Cognitive Computation*, vol. 9, pp. 423–435, 2017.

[2] Ai challenger 2018. [Online]. Available: https://challenger.ai/competition/fsauor2018

[3] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.

[4] E. Boiy and M. Moens, "A machine learning approach to sentiment analysis in multilingual web texts," *Information Retrieval*, vol. 12, no. 5, pp. 526–558, 2009.

[5] R. P. . S. H. Manning, C. D., *Introduction to information retrieval*. Cambridge University Press, 2008, ch. 13.

[6] H. M. Wang, Y. and L. Zhao, "Attention-based lstm for aspect-level sentiment classification," *Proceedings of the 2016 conference on empirical methods in natural language processing*, pp. 606–615, 2016.

[7] C. E. Peng, H. and A. Hussain, "A review of sentiment analysis research in chinese language," *Cognitive Computation*, vol. 9, no. 4, pp. 423–435, 2017.

[8] Y. Song, S. Shi, J. Li, and H. Zhang, "Directional skip-gram: Explicitly distinguishing left and right context for word embeddings," *NAACL (Short Paper)*, 2018.

[9] Cs 224d course material. [Online]. Available: https://cs224d.stanford.edu/lecture_notes/notes1.pdf

[10] W. Gong. (2017) Chinese word vectors. [Online]. Available: https://primer.ai/blog/Chinese-Word-Vectors/

[11] Ai challenger 2018 baseline. [Online]. Available: https://github.com/AIChallenger/AI_Challenger_2018

[12] C. Olah. (2015) Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/