Project Proposal: Optimization of Sentiment Analysis using Embedding

Shuwen Qiu
005061171

Dongyun Zhang
004844633

Yanzhao Wang
405229892

Zhuyan Chen
705219448

1 PROBLEM AND GOALS

Sentiment analysis is a field within natural language processing(NLP), that identifies, extracts and quantifies the subjective information in the text. Nowadays, sentiment analysis is widely used in social media monitoring, customer service, and other business analytics.

In the sentiment analysis pipeline, embedding method has been constantly used as a way of representing vocabularies and texts. Serving as an approach of preprocessing the text data for classifier input, this method has been proved to be very effective. Several embedding methods have become very popular, for example, Word2Vec, GloVe, and fastText. Meanwhile, the performance of embedding methods also varies with contents. In our project, we are going to explore the most effective word embedding method under our sentiment analysis models. As of sentiment analysis, we will focus on polarity classification of the yelp comments, predicting whether the post is positive or negative. Multiple methods will be tried, including both simpler and basic machine learning models, such as logistic regression, SVM, and advanced and more complicated models, such as MLP.

Our goal is to compare multiple word/sentence embeddings along with several sentiment analysis classifiers and come out the best-performance sentiment analysis model with the assistant of the embedding technique.

2 SOLUTION PLAN

2.1 Part I

(1) **Word Embedding**

(a) **Bag-of-words with N-grams representation**
The most common and intuitive way is to represent words as atomic symbols or "one-hot" vectors. However, problems include that the dimension can be too large, the vector is often sparse, and new words cannot be represented.

(b) **Vector Representation**
GloVe GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.

FastText FastText provides two models for computing word representations: skipgram and cbow ('continuous-bag-of-words'). The skipgram model learns to predict a target word thanks to a nearby word. On the other hand, the cbow model predicts the target word according to its context. The context is represented as a bag of the words contained in a fixed size window around the target word.

(2) **Sentence Embedding**

(a) **RNN**: The chain-like nature of recurrent neural networks allow them to solve problems related to sequences and lists, as they can persist information and connect previous information to the present task. However, a common problem is that if the gap between the relevant information and the point where it is needed to becomes very large, RNNs become unable to learn to connect the information.

(b) **LSTM**: LSTM is a special kind of RNN. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTMs are explicitly designed to avoid the long-term dependency and vanishing gradient problems that can be encountered when training traditional RNNs. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The activation function of the LSTM gates is often the logistic function.

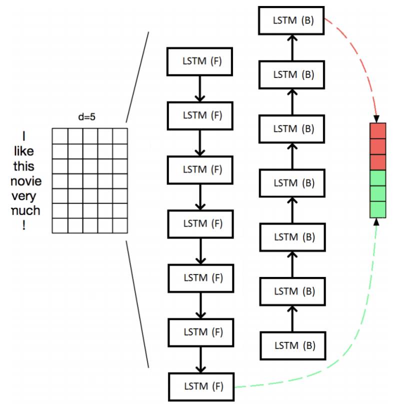

Figure 1: LSTM[1]

(c) **CNN**: The input of the network are the sentences, which are tokenized into words. Each word is mapped to a word vector representation, i.e. a word embedding, such that an entire sentence can be mapped to a matrix of size $s \times d$, where s is the number of words in the tweet and d is the dimension of the embedding space. Then several convolution operations of various sizes are applied to this matrix. A single convolution involves a filtering matrix $w \in R^{h \times d}$ where h is the size of the convolution, meaning the number of words it spans. The max-pooling operation extracts the most important feature for each convolution, independently of where in the sentence this feature is located. CNN's structure effectively extracts the most important n-grams in the embedding space, which is why these systems are good at sentence classification.

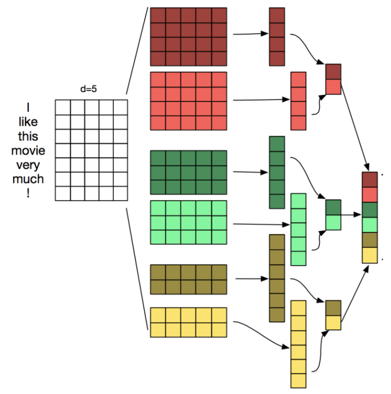

Figure 2: CNN[1]

2.2 Part II: Classifiers

As illustrated below, we plan to adopt one linear classifier(Logistic regression), two non-linear classifiers(kernel SVM and MLP). For the neural network, we also adopt the transfer learning method to further improve the performance.

(1) **Logistic Regression**. Suppose that x is the embedding of a sentence. The probability that x is classified into class c can be represented as: $p(c|x) = sigmoid(\beta^T x)$

(2) **Kernel Support Vector Machine(SVM)**. The objective of SVM is to find the hyperplane with the biggest margin to separate data points. In order to break the limit of linear classifiers, kernel methods are introduced so that data points can be better separated in the kernel space. In this task, we apply the Gaussian Kernel. $K(x_i, x_j) = exp(||x_i - x_j||^2/\sigma^2)$

(3) **Multi-layer perceptron(MLP)**. The input of the MLP is the embedding of a sentence, and the output is the probabilities that the input belongs to each class. For each layer, the output is a linear combination of the input after adopting an activate function, and the output of the last layer will be the input of the next layer. We adopt cross-entropy as the loss function, ReLU as the activate function and SGD as the optimizer following the design in [3].

(4) **Fine-tuning**. Models trained on small datasets usually face the problem of overfitting. In order to avoid this problem, a pre-trained language model can be first build using a big dataset, and then the model can be fine-tuned on a small dataset for a specific task. Following the instruction in [2], we apply the transfer learning method to a model pre-trained on a large dataset Wikitext-103[5], and then fine-tune on sentiment analysis task.

3 DATA PLAN

3.1 Dataset

The datasets that we plan to evaluate our model with are as follows:

(1) **Yelp Reviews Dataset**. This dataset is collected from a Kaggle competition on Natural Language Processing and Sentiment Analysis. It contains more than 5,200,000 reviews on restaurants and business listed on Yelp. The label of each review are stars from 1 to 5, and we form it into a binary classification problem by defining positive review as reviews with more than 3 stars and negative reviews as those with only 1 start. The dataset also has features including cool column, useful column and funny column (which are the number of cool, useful, funny votes the review received). We will examine whether these features will improve our model's performance on this large dataset.

Additional tentative datasets

(2) **Large Movie Review Dataset**. This dataset [4] collects 50,000 reviews on movies from IMDB, and the 50,000 reviews are split evenly into 25,000 train sets and 25,000 test sets. Each review is assigned with a score from 1 to 10, where the review with score ≤ 4 are classified as negative and those with score ≥ 7 are classified as positive. The review with neutral ratings are not included in the dataset and we are going to evaluate our model's performance on binary sentiment classification with it. This dataset is maintained by Stanford AI, with raw text and already processed bag of words formats provided.

(3) **Multi-domain Sentiment Analysis Dataset**. This dataset [6] features product reviews from Amazon and is maintained by the Johns Hopkins University's Department of Computer Science. Each review contains a star rating from 1 to 5, and with this dataset, we plan to evaluate our model's performance on multi-class sentiment classification. The reviews in this dataset are taken from multiple product types (domains), with some domains having hundreds of thousands of reviews while some domains only have a few hundreds of reviews. We will also explore our model's performance on various product review domains and see whether the number of training data makes a difference.

3.2 Data Processing

In this project, we plan to process our data in the workflow as follows:

(1) **Preprocessing**. The first step is to preprocess the data, mainly includes removing noisy data and unmeaningful features in the training sequences. Features such as review ID, review

date, user ID should be removed. We will also preprocess the labels and form a binary classification problem by creating positive and negative labels and throwing out neutral reviews.

(2) **Tokenization**. We plan to use python's Natural Language Toolkit (NTLK) to tokenize the text for each review. Possible tokenization techniques include stop words removal, punctuation removal, and keyword analysis.
(3) **Word Embedding**. Use the Word Embedding model we designed to transform the tokenized text into vector/matrix/graph data.
(4) **Normalization and Standardization**. Perform possible normalization and standardization on the embedded word features.

4 EVALUATION PLAN

In this project we plan to evaluate our model with the following metrics:

(1) **Accuracy**: The rate our model correctly predicts the review's label. The formula in a binary classification task is Accuracy = $\frac{true\ positives + true\ negatives}{total\ examples}$
(2) **Precision and Recall**: In a binary classification task, we will use precision and recall to measure the true positive rate in ground truth labels and predicted labels.
Precision = $\frac{true\ positives}{true\ positives + false\ positives}$
Recall = $\frac{true\ positives}{true\ positives + false\ negatives}$
(3) **F1-Score**: F1-Score = $2 * \frac{precision * recall}{precision + recall}$, which is the harmonic mean of precision and recall. In a multi-class classification problem we will use weighted F1-Scores.
(4) **Visualization**: Visualize the classification results with histograms and confusion matrices. We will also seek the possibility to visualize the results into wordcloud and intuitively find out the key words in each class.

Figure 3: A wordcloud example

5 SCHEDULE

Week 4 - 6: Related paper searching and reading. Summarize the existing word embedding and sentiment classification models. After group meeting and brief discussion, methods implementation will be started.

Week 6 - 8: Continue methods implementation, at the same time, tuning the parameter of each model and identify several evaluation metrics that can comprehensively evaluation the major design of our project. Since our word embedding and classifier are implemented separately, in order to combine them for evaluation, we might need to spend time to do some code review.

Week 8 - 10: Model evaluation and analysis, small changes in models are expected. During this period, we will also focus on the preparation of the report, and final presentation.

6 WORK DIVISION

We will try to make the work evenly distributed among all members. For now, our code implementation will focus on two major parts, word embedding and sentiment classifier. The word embedding part will be involved with at least four different approaches. Shiwen, Dongyun, Yanzhao, and Zhuyan will in charge of GloVe, FastText, LSTM and CNN separately.

At least four sentiment classifiers will be implemented. According to the plan, Shiwen, Dongyun, Yanzhao, and Zhuyan will in charge of Multi-later perceptron(MLP), fine-tuning transfer learning method, logistic regression, and SVM separately. More models will be added, and some ideas mentioned above are subjected to change.

All group member will participate in group meetings and contribute to project proposal group presentation and report.

REFERENCES

[1] Mathieu Cliche. 2017. BB_twtr at SemEval-2017 task 4: twitter sentiment analysis with CNNs and LSTMs. *arXiv preprint arXiv:1704.06125* (2017).
[2] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).
[3] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130* (2017).
[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning Word Vectors for Sentiment Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 142–150. http://www.aclweb.org/anthology/P11-1015
[5] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843* (2016).
[6] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2013. Parsing With Compositional Vector Grammars. In *EMNLP*.