573 Project

Yu Ying Chiu, **Jiayi Yuan**, **Yian Wang**, **Chenxi Li**
Department of Linguistics
University of Washington
Seattle, WA, USA
{kellycyy, jiayiy9, wangyian, cl91}@uw.edu

Abstract

This paper presents a system for emotion and personality detection in open-domain dialogue - the FETA-Friends dataset which contains transcripts for all 10 seasons of the TV show Friends. We choose emotion detection as the primary task and personality detection as the secondary task. For the emotion detection task, the dataset includes utterances with annotated emotions, and the evaluation is based on the micro-F1 and W-F1 scores. For the personality detection task, the dataset includes short conversations with annotated binary personality traits, and the evaluation is based on accuracy. Our system uses BERT encoding and a classification layer for prediction, and includes a lexicon baseline model for comparison

1 Introduction

Large language models such as BERT have shown remarkable success in various natural language processing tasks. However, leveraging these models for downstream tasks such as sentiment analysis often requires fine-tuning on specific datasets. In this work, we operates on the FETA-Friends dataset, which contains transcripts for all 10 seasons of the TV show Friends, and presents a system that does emotion recognition and personality detection on this dataset.

2 Task description

We select the shared task (FETA challenge) by Albalak et al. (2022). It is a new benchmark for few-sample task transfer in open-domain dialogue. In this benchmark, we selected the FETA-Friends as our dataset from Chen and Choi (2016). It involves transcripts for all 10 seasons of the TV show (Friends).

The FETA-friends benchmark contains 7 tasks. We selected Emotion Recognition (Emory NLP) by Zahiri and Choi (2017) as our primary task and Personality Detection by Jiang et al. (2020) as our adaptation task.

For the primary task, the dataset contains utterances with one of the annotated emotions (eg. Neutral, Joyful, Powerful, Mad, Scared etc.). According to the dataset github [1], the dataset fields include the utterance id, speaker name, transcript, tokens and annotated emotion. It is an utterance-level classification task. The evaluation is to calculate the micro-F1 and W-F1 scores of the prediction (classification output).

For the adaptation task, the dataset contains short conversations with annotated binary Big Five personality traits (Agreeableness, Conscientiousness, Extroversion, Openness, and Neuroticism). According to the dataset github [2], each personality trait was annotated on a scale of -1, 0 , 1. The annotation of each trait in a short conversation will be summed up. The dataset contains scene id, character, AGR (Agreeableness), CON (Conscientiousness), EXT (Extroversion), OPN (Openness), NEU (Neuroticism), and text. It is a dialogue-level classification task. The evaluation is to calculate the accuracy of the prediction output.

For the key dimensions, the primary task has emotion as affect type, classification as recognition type, TV transcript as Genre, aspect-specific emotion as the target, text as modality, and English as language. For the adaptation task, it has personality type as affect type, aspect-specific personality detection as target, and all the other dimensions are the same as the primary task selected.

The FETA benchmark [3] provides access to data (train, test, dev), training and evaluation tool as a reference. The original author of dataset [4] also

[1] https://github.com/emorynlp/emotion-detection
[2] https://github.com/emorynlp/personality-detection
[3] https://github.com/alon-albalak/TLiDB/blob/master/FETA_README.md
[4] https://github.com/emorynlp

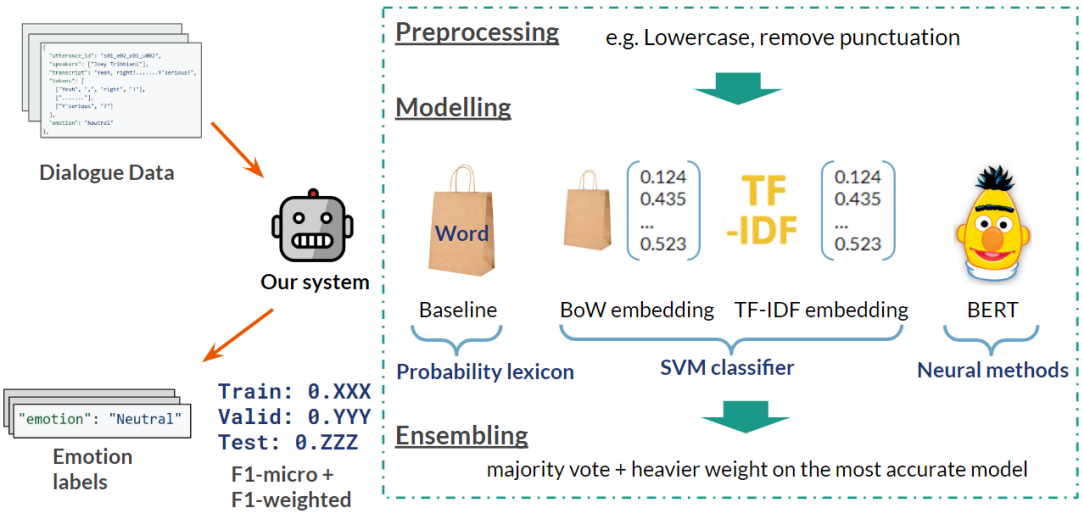

Figure 1: System Overview of emotion detection (ensembling method will be completed in D4)

provides access to the same dataset.

3 System Overview

From Figure 1, our system aims to predict the emotion labels from 7 classes (joyful, power, peace, neutral, sad, scared, mad) from the dialogue data and generate the prediction metrics on different datasets.

The baseline for our system is a bag-of-words approach that uses lexicons to detect emotions. It creates a dictionary of lexicons and emotional labels by iterating through the training data. The frequency of each lexicon for each emotional label is computed and used to predict the emotions of utterances in the dev and test set based on the sum of the probabilities of each lexicon in the utterance.

One model we implemented is an SVM classifier. The data is extracted and appended to a list, and then transformed into the embeddings. The embeddings and their corresponding labels are fed into an SVM classifier with a linear kernel for training.

Apart from SVM, we also implemented a neural network model. The system firstly load the dataset and preprocesses it by adding the required annotation specified by the command, and transformed into the specific json format data. During training, it encodes the original transcript using pretrained BERT model[5], and a classification layer is added in the end to make the predication. The parameters on the classification layer is trained uses Adam optimization algorithm to update the network weight each training epoch, and repeats for the number of epochs defined by the command, default to be 10 epochs. After training, evaluation is done against test set. In the end, it outputs the metrics and the prediction files. It also saves the trained model to the saved model path to allow future evaluation on the same model.

The provided command ran evaluation on the tree models as described above.

4 Approach

4.1 Baseline - lexicon model

We implement a lexicon model as the baseline. It is a simple bag-of-words probability method that uses lexicons to detect emotion. We will first extract a lexicon dictionary from the training data. The dictionary has two keys, the first key is a lexicon, the second key is the emotional label, and the value is its frequency. For example, word_dict['like']['joyful'] = 50, word_dict['like']['peaceful'] = 20. To obtain the dictionary, we will iterate through the training data. For every utterance in the training data, we will separate it into tokens. Then we will add one frequency for the utterance's true label to every token in this utterance.

After we obtain the dictionary, we will use it to make prediction for dev and test set. For every utterance in dev and test set, we will separate it into tokens, and for each token, we will compute its possibilities based on frequency. The possibility for an utterance is computed as the sum of the possibilities of its tokens. The utterance will be predicted as the label with the highest possibility.

[5] https://huggingface.co/bert-base-uncased

4.2 BERT model

We implemented another model based on BERT (bert-base-uncased). Adam has been used as its optimizer and 3e-5 as the learning rate of our model, We input the original dataset without any prepossessing to test for the base BERT model performance. The task is a multi-class classification task. Our model contains linear neural network layers without restriction of maximum length and with the number of possible classes. The input data is tokenized and passed through those layers. A classification layer is added at the end for outputting the classified emotion types.

We trained the model for the train data (8629 examples) provided for 10 epochs. Then we evaluated the model with dev data (2065 examples) for 10 epochs. For testing, we used the test data (110 examples) for comparison.

4.3 SVM model

We implemented SVM as our third model, the results of which will be compared with other models and be used to form ensemble.

Unlike the previous two models that directly extract tokens from the original data, in this model, we extracted the transcript - which is of type string - and append them to a list. Then we transformed all sentence into two types of embeddings of equal length: bag of words embedding and Tfidf embedding. Particularly, we used Sklearn package to achieve the transformation. Lastly, we fed the embeddings and their corresponding labels to an SVM classifier with a linear kernel, again using Sklearn, for training, and achieved the following score presented in the results section.

4.4 Ensemble

We plan to implement an ensemble method for D4. The ensemble method will combine three models: lexicon model, SVM model with bags-of-words embedding, and SVM model with Tfidf embedding. We adopt a majority vote strategy. If any two of the three methods predict the same label, the ensemble method will predict this label. However, suppose the three methods predict three different labels, and a majority vote doesn't exist. In that case, the ensemble method will predict the same label as the SVM model with Tfidf embedding since it is the model with the highest F1-micro score.

5 Results

The task suggested using two metrics (F1-micro and F1-weighted). The difference between F1-micro and F1-weighted is the consideration of proportion of labels in the dataset. We decided to select the F1-micro as our main metric comparison since we believed that human utterances' emotion label distribution is not even (i.e. some emotions is more commonly appeared). Using F1-micro may boost the prediction performance better even with relatively more predictions on one class of emotion labels. Both F1-micro and F1-weighted will be reported in the following sections

5.1 Baseline - lexicon model

The model attained 0.3016 as F1-micro and 0.2302 as F1-weighted on dev set.

5.2 BERT model

For training on the 10th epoch, the model attained 0.9735 as F1-micro and 0.9734 as F1-weighted. For evaluation on the 10th epoch, the model attained 0.3705 as F1-micro and 0.3664 as F1-weighted. For testing, the model attained 0.4189 as F1-micro and 0.3913 as F1-weighted.

5.3 SVM model

The SVM model with bags of words embedding attain 0.3554 as F1-micro and 0.3174 as F1-weighted on dev set. The SVM model with Tfidf embedding attain 0.3714 as F1-micro and 0.2990 as F1-weighted on dev set.

5.4 Modification - BERT model hyperparameter tuning

Our default model is a bert-base-uncased model with number of epoch as 10, learning rate as $3e-5$, optimizer as Adam and effective batch size as 60. Due to the limited GPU resources and the huge training dataset size, we did the hyperparameter tuning one by one based on the above default setting. The subsections below showed the metrics on validation set. Due to the randomness nature of neural network model, we repeated 5 sets of experiments per hyperparameter tunning and reported the average and standard deviation for fair comparison.

5.4.1 Number of epoches

It means the number of times of the entire dataset passing the algorithm. We tested 5, 10 and 20 epoches.

epoch	F1-micro	F1-weighted
5	0.36975 (0.0148)	**0.368825** (0.0094)
10	0.36494 (0.018)	0.36286 (0.0142)
20	**0.3735** (0.0131)	0.36665 (0.0092)

Table 1: Metrics for hyperparameter tuning on number of epoches. It reported average of five experiments with its standard deviation in the bracket.

In Table 1, we found that the F1-micro for 20 epoches is the highest while the F1-weighted for 5 epoches is the highest among all. Due to the aforementioned reason, we considered the F1-micro metric and hence we selected 20 as the number of epoches.

5.4.2 Learning rate

It means the speed at which the model learns by contrlling the amount of apportioned error during weight updates. We tested 1E-5, 3E-5 and 5E-5.

rate	F1-micro	F1-weighted
1E-5	**0.37205** (0.01033)	0.36185 (0.0103)
3E-5	0.36494 (0.018)	**0.36286** (0.0142)
5E-5	0.363925 (0.018)	0.361825 (0.0138)

Table 2: Metrics for hyperparameter tuning on learning rate. It reported average of five experiments with its standard deviation in the bracket.

In Table 2, we found that the F1-micro for 1E-5 is the highest while the F1-weighted for 3E-5 is the highest among all. Due to the aforementioned reason, we considered the F1-micro metric and hence we selected 1E-5 as the learning rate.

5.4.3 Optimizer

It means changing the optimization algorithm for gradient descent of the model. We compared Adam and AdamW.

optimizer	F1-micro	F1-weighted
Adam	0.36494 (0.018)	0.36286 (0.0142)
AdamW	**0.3792** (0.0109)	**0.36973** (0.0072)

Table 3: Metrics for hyperparameter tuning on optimizer. It reported average of five experiments with its standard deviation in the bracket.

In Table 3, we found that AdamW attains the highest F1-micro and F1-weighted.

5.4.4 Effective batch size

In our model, the gradient accumulation step is calculated by $\frac{\textbf{effective batch size}}{\textbf{gpu batch size}}$. We changed the effective batch size with 20, 40, 60.

batch size	F1-micro	F1-weighted
20	0.371175 (0.003)	0.361675 (0.0035)
40	**0.373375** (0.0149)	**0.36295** (0.0152)
60	0.36494 (0.018)	0.36286 (0.0142)

Table 4: Metrics for hyperparameter tuning on effective batch size. It reported average of five experiments with its standard deviation in the bracket.

In Table 4, we found that effective batch size as 40 attains the highest F1-micro and F1-weighted.

5.4.5 Result of combination of best hyperparameter

Based on the above trials, we selected bert-base-uncased model with learning rate of 1E-5, AdamW as optimizer, 40 as effective batch size. We did 10 epochs instead due to the lack of training time (it takes more than 5 hours on our current setting T4 GPU). We aim to repeat this experiment with 20 epochs and provide its checkpoint on D4. The F1-micro attained is 0.3937 and the F1-weighted is 0.3776 on validation set. It increased by 6.26% on F1-micro and 3.57% on F1-weighted.

6 Discussion

As shown in the results section, the F1 score for both the baseline model and the BERT model have large improvement space. We believe the low score of the BERT model is largely due to different domains: BERT is known to be trained from formal texts like books, but our task is to classify emotions in a more informal setting - daily dialogues. The baseline model has several disadvantages as well. First, the baseline method particularly extracts transcription from a single turn, ignoring everything before and after the current turn. The disadvantages of only using the current turn is that it did not use the contextual information from the previous turns. However, when we actually incorporate the previous turns into the baseline method, the accuracy actually drops, because the lexical method gives the same weight for words appeared in the current turn and the previous turns. Second, the baseline method mainly relies on lexical components of the

utterance, and it doesn't take full advantage of the semantic components of the dialogue as an entity.

6.1 Comparison between different models

Method	F1-micro	F1-weighted
Baseline BoW	0.3016	0.2302
SVM (BoW embedding)	0.3554	0.3174
SVM (TF-IDF embeddding)	0.3714	0.2990
BERT (tuned)	**0.3937**	**0.3776**

Table 5: Comparison between different models on validation set.

In Table 5, we implemented different models and our best model is BERT after hyperparameter tuning. Our performance improved from 0.3016 to 0.3937 by 30.53% on F1-micro while improved from 0.2302 to 0.3776 by 64.03% on F1-weighted when compared with our baseline model. We noticed that the F1-micro metrics for all models are lower than their F1-weighted respectively. Due to the proportion problem mentioned in Section , we considered F1-micro as our main metric. The result in this subsection also showed that the uneven distribution of emotion label of the datasets. It may further prove our hypothesis (i.e. the reason of choosing F1-micro) that the emotion label distribution is uneven as well in our real existing world.

6.2 Other attempts

After D2, we planned to incorporate a more sophisticated lexicon method in our model either by leveraging existing lexicons or creating a new lexicon specifically for our data. These two ideas turned out to be unfit for our data. This section is a detailed explanation on what we tried and why they did not work.

6.2.1 Leveraging existing lexicons

The first attempt was to use existing emotion lexicons such as NRC Word-Emotion Association Lexicon (EmoLex), a lexicon that has assigned emotion values to the vocabularies through crowdsourcing. The immediate problem was that the our training data contains seven labels, three of which ('Neutral', 'Powerful', and 'Peaceful') are absent in EmoLex. Such a discrepancy between two sets of labels forced us to seek advice from the instructor. Combining suggestions and our own ideas, we planned to design a metric that represents the missing label with availale labels in a lexicon. For instance, 'Neutral' label could equal 'Positive' plus 'Negative' in EmoLex. Or alternatively, we could still use Emolex to obtain the emotion values, and directly feed them to a machine learning method - the machines might still learn from the data after all. We decided to not proceed with the two options when we discovered that many tokens in our training data were absent in EmoLex. To be specific, the data in our task is daily conversation transcripts that contain many input sentences such as 'Oh, yeah' or just 'Yeah...', and we simply cannot get any emotion values regarding these words from EmoLex.

6.2.2 Creating a lexicon

We also thought about creating a lexicon on our own. Given the limited time and budget, we planned to use the traditional way of selecting seeds and expanding lexicon rather than crowd-sourcing, but this idea was soon overthrown as we knew more about our data. As mentioned previously, our data is highly conversational, so manually creating seeds for each label might not be an optimal start. For instance, intuitively, we would create a seed 'joy' for label 'joyful', but the token 'joy' only appeared once in the whole training data. Thus, we resorted to a more data-driven approach - selecting tokens that have the highest occurrence under each label as the seeds. The results were again undesirable as the tokens of highest frequency were unexpectedly punctuation or discourse markers such as 'oh' and 'umm'.

References

Alon Albalak, Yi-Lin Tuan, Pegah Jandaghi, Connor Pryor, Luke Yoffe, Deepak Ramachandran, Lise Getoor, Jay Pujara, and William Yang Wang. 2022. FETA: A benchmark for few-sample task transfer in open-domain dialogue. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10936–10953, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yu-Hsin Chen and Jinho D Choi. 2016. Character identification on multiparty conversation: Identifying mentions of characters in tv shows. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pages 90–100.

Hang Jiang, Xianzhe Zhang, and Jinho D Choi. 2020. Automatic text-based personality recognition on monologues and multiparty dialogues using attentive networks and contextual embeddings (student abstract). In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13821–13822.

Sayyed M Zahiri and Jinho D Choi. 2017. Emotion detection on tv show transcripts with sequence-based convolutional neural networks. *arXiv preprint arXiv:1708.04299*.