University of Science and Technology of Hanoi
Information Communication Technology Department

Natural Language Processing

Emotion Recognition for Vietnamese Social Media Text Using Language Model

Group 4

Nguyen Trong Minh	22BI13304
Nguyen Manh Hung	22BI13183
Le Xuan Loc	22BI13256
Tran Luong Hoang Anh	22BI13039
Cao Hieu Vinh	22BI13470
Dinh Tuan Kiet	22BI13229

Prof: Nghiem Thi Phuong

March,2025

1 UIT-VSMEC Dataset

1.1 Overview

The UIT-VSMEC (Vietnamese Social Media Emotion Corpus) is a high-quality dataset designed for emotion recognition in Vietnamese text, particularly in the context of social media. It consists of 6,927 manually labeled sentences, each assigned to one of six basic emotion categories:

- Enjoyment
- Sadness
- Anger
- Fear
- Disgust
- Surprise

This dataset was developed to support research in Natural Language Processing (NLP), with a focus on understanding and analyzing emotions in Vietnamese social media texts. Each sentence was labeled based on its expressed emotion, following a well-defined annotation guideline. The source texts were collected from various online platforms, including social media posts, comments, and discussions, making the dataset highly representative of real-world user-generated content.

1.2 Average Sentence Length by Emotion

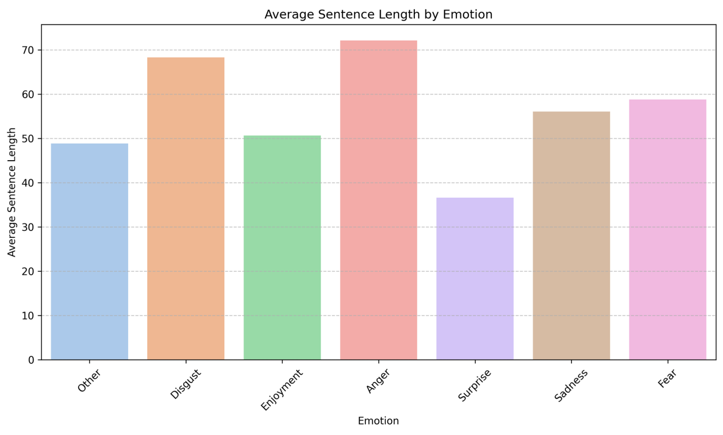

Figure 1: Average Sentence Length by Emotion

The bar graph in Figure 1 illustrates the average sentence length for each emotion category. As we can see, sentences are expressed as **Anger** and **Disgust** accounts for a high proportion, averaging over 65 words, while **Surprise** has the shortest sentences. This suggests that emotions like **Anger** and **Disgust** may require more context to express.

1.3 Emotion Class Distribution

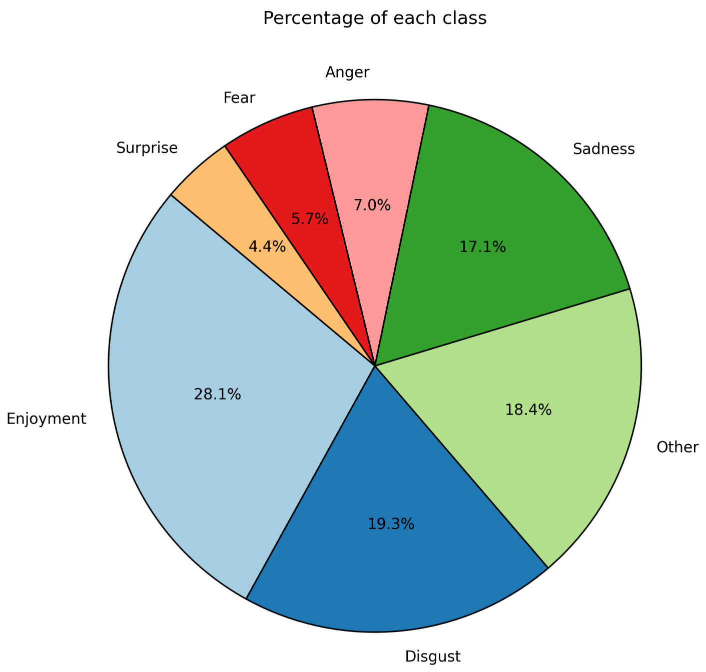

Figure 2: Emotion Class Distribution

According to the Figure 2, the pie chart displays the distribution of emotions within the dataset. **Enjoyment** is the most prevalent category, about 28.1% of the dataset, while **Surprise** is the least frequent, only 4.4%. This imbalance may cause the model to perform better for high frequency emotions while struggling with lower-frequency onces. The others emotion class in the range of 5%-20%

2 Methodology

In this work, we performed experiment on 3 approaches for emotion classification.

- **Standard sequence classification**: Model with embedding backbone and a dense classification head.
- **Constrastive classification**: Model with embedding backbone, encode both input sequence and output class, decide by cosine similarity between input sequence and out put class.
- **LLM prompting**: We utilize the power of LLM, in this case is Qwen2.5, to predict emotion.

2.1 Input format

2.1.1 Standard sequence classification

In this approach, models take in original sequence, encode it to a vector. Each class is encoded to numerical format from 0 to 6.

Example:

```
công nhận sáng tạo thật đấy                              - Enjoyment (2)
cho mình xin bài nhạc tên là gì với ạ                    - Others (4)
cho đáng đời con quỷ . về nhà lôi con nhà mày ra mà đánh - Disgust (1)
```

2.1.2 Constrastive classification

This approach takes form by aligning sentence embeddings (from text input) with embeddings of emotion labels (as full strings) in a shared vector space, turning the task into word embedding with similarity calculation instead of a classifier and preserving the raw string input of target emotion.Allowing further semantics understanding as the model now can understand what the emotions are.

For this approach to work well, we also translated label strings to Vietnamese.

Example:

```
công nhận sáng tạo thật đấy                              - Thích thú
cho mình xin bài nhạc tên là gì với ạ                    - Khác
cho đáng đời con quỷ . về nhà lôi con nhà mày ra mà đánh - Kinh tởm
```

2.1.3 LLM Classification

A prompt-based supervised fine-tuning approach with Qwen2.5 is used to train the model on emotion classification. The template is formatted as follows:

```
Classify the emotion in the sentence below.

### Sentence:
{}

### Emotion:
{}
```

Where in ”Sentence” and ”Emotion” represent the sentence and the corresponding emotion from the dataset. The entire prompt is then tokenized accordingly.

2.2 Output

2.2.1 Standard sequence classification:

The output is a normal numerical logits vector of each class. We use `argmax()` to get what class is predicted for the corresponding sequence.

Example:

```
Sentence: ước gì sau này về già vẫn có thể như cụ này :))
Logits vector: [-0.9160298  -0.56967443  2.5012646  -0.7669704 0.14470923
-0.3288063 -0.10403331]
Class predicted: 2 ~ Enjoyment
```

2.2.2 Contrastive Classification:

The output is a normal numerical cosine similarity vector, each element is the similarity between the input sequence with the label (in Vietnamese). We then use `argmax()` to get final class prediction output. Example:

```
Sentence: per nghe đi rồi khóc 1 trận cho thoải mái . đừng cố gồng mình
lên nữa
Similarity vector: [0.3300 0.0883 0.1186 0.3033 0.1283 0.3483 0.2857]
Class predicted: 5 ~
```

2.2.3 LLM Classification:

LLM generate an output similar to the input prompt but add predicted emotion at the end, after "`### Emotion:`".

```
Classify the emotion in the sentence below.
### Sentence:
tôi đang rất cáu !!!
### Emotion:
Anger<EOS_TOKEN>
```

The emotion label is extracted from this output using regular expression, and give the final result is "Anger".

3 Obtained outcome

We perform finetuning on 3 models, 2 embedding models: Vietnamese-bi-encoder (VBE), GTE-multilingual-base (GTE), and 1 LLM model Qwen2.5-7B

Table 1: Evaluation results on UIT-VSMEC dataset.

Models	Accuracy (%)	F1-Score (%)	Precision(%)	Recall(%)
GTE standard	53.82	48.9	47.38	53.82
GTE contrastive	35.93	32.48	34.83	35.93
VBE standard	61.9	**61.86**	**62.24**	**61.9**
VBE contrastive	58.73	57.69	56.94	58.73
Qwen2.5-7B	**69.9**	-	-	-

From this table, we can see that Qwen 2.5 achieved exceptional results, with accuracy reaching 69.9%. However, due to limited hardware resources, we were able to perform the training once, hence we can only retrieve the accuracy . Generally, the BKAI model performs better than the Alibaba model because it has been optimized for Vietnamese.

4 Tutorial

4.1 Access our work

You can access the github repository via the following Github repository, and the dataset via huggingface dataset

4.2 Documentation

This documentation explain what are inside our github repository.

4.2.1 Flow

- Standard Classification Fine-Tuning:
  1. Load dataset, encode label to id.
  2. Load model, tokenizer from huggingface, or cached folder.
  3. Use Trainer class from transformers library to finetune.
  4. Evaluate fine-tuned model.
- Contrastive Classification Fine-Tuning:
  1. Load dataset, encode label to vector representations.
  2. Define custom loss function from cosine similarity.
  3. Train with AdamW optimizer.
  4. Evaluate fine-tuned model.
- LLM Classification:
  1. Load model, tokenizer, wrap with PEFT model from unsloth.
  2. Load dataset, format with template prompt.
  3. Use SFTTrainer class from trl library to train.
  4. Evaluate fine-tuned model.

4.2.2 Configuration

You can access our default config files in `config` folder. There are 3 default config files for 3 models with their corresponding name.

For embedding model, there are 2 config parts for contrastive classification and standard classfication.

Standard Classification:

1. Train: These are hyperparameters in `TrainingArguments` for `Trainer` class for training.

```
"eval_strategy": "no",
"save_strategy": "epoch",
"learning_rate": 2e-5,
"per_device_train_batch_size": 8,
"num_train_epochs": 1,
"weight_decay": 0.01,
"logging_steps": 100,
"push_to_hub": false,
"report_to": "none"
```

2. Tokenizer: These are parameters set for tokenizer to transform input sequences in dataset.

```
"truncation": true,
"padding":"max_length",
"max_length": 218
```

3. Evaluate: These are hyperparameters in `TrainingArguments` for `Trainer` class for evaluation.

```
"per_device_eval_batch_size": 1,
"do_train": false,
"do_eval": true,
"eval_strategy": "epoch"
```

Contrastive Classification

1. Train: These hyperparameters are used for training setting in our work only.

```
"batch_size": 16,
"shuffle": true,
"epochs": 1,
"learning_rate": 2e-5
```

2. Tokenizer: There is only one parameter for max length of sequence.

```
"max_length": 258
```

3. Evaluate: There is only one parameter for batch size, this should be 1.

```
"batch_size": 16,
"shuffle": true,
"epochs": 1,
"learning_rate": 2e-5
```

LLM Classification

1. Peft model: These are parameter for function `FastLanguageModel.get_peft_model()`

```
"r": 16,
"target_modules": ["q_proj", "k_proj", "v_proj", "o_proj",
"gate_proj", "up_proj", "down_proj"],
"lora_alpha": 16,
"lora_dropout": 0,
"bias": "none",
"use_gradient_checkpointing": "unsloth",
"random_state": 3407,
"use_rslora": false,
"loftq_config": null
```

2. Train: These are hyperparameters in `TrainingArguments` for `SFTTrainer` class for Supervised Finetuning.

```
"per_device_train_batch_size": 2,
"gradient_accumulation_steps": 4,
"warmup_steps": 5,
"num_train_epochs": 1,
"learning_rate": 2e-4,
"logging_steps": 1,
"optim": "adamw_8bit",
"weight_decay": 0.01,
"lr_scheduler_type": "linear",
"seed": 3407,
"report_to": "none"
```

4.2.3 Running

To recreate our work, you must clone our github repository, install every required libaries in `requirements.txt`. After that you can run `main.py` with python 3.12.7. We support command line arguments for each tasks.

```
--model [
    'vbe' : vietnamese-bi-encoder,
    'gte' : gte-multilingual-base ,
    'qw7' : Qwen2.5-7B,
    'your-checkpoint-folder' : Your path to saved-folder
]
--task [
    'ftstd' : standard classfication finetuning,
    'ftsim' : contrastive classification finetuning,
    'evstd' : standard classfication evaluation,
    'evsim' : contrastive classification evaluation,
]
--config : path to json config file, if not provided, python will
    select 1 of 3 default config files base on model name. You must
    manually select config file when evaluate model.
--save-folder : path to save model after training, if not provided,
    it will create a name like this model-task-date-month-hour-minute.
    You need to manually select folder when evaluate model. Else, it
    will fail to save confusion matrix.
```

Example of a training command line:

```
# train Qwen2.5

python main.py --model qw7 --task ftllm
```

Example of an evaluating command line:

```
# Evaluate vietnamese-bi-encoder for standard classification task

python main.py --model vietnamese-bi-encoder-ftstd-273339 --task evstd
--config config/vietnamese-bi-encoder.json --save-folder
vietnamese-bi-encoder-ftstd-273339
```