Data mining Final

Created	@2024年11月27日 下午1:16
Class	DM
學生	林靖淵 113356040

Kaggle Score

Rank

110	▲ 6	TMU b507110014		0.34566	6	5d
111	—	vope1230		0.34522	6	6d
112	—	hiwei6		0.33476	3	2d
113	▲ 6	jingyaun_nccu		0.33396	5	11d
114	▲ 4	VictorKuang_lhu		0.33328	1	1d
115	▼ 1	Brady Huang		0.33327	4	17h
116	▼ 3	eating chew		0.33289	2	1d
117	—	Maojack		0.32774	2	2d

Submission record

submission_decision_tree_2.csv Complete · 11d ago	0.21566	0.22333	☐
submission_decision_tree_2.csv Error · 11d ago			
submission_DL_tfidf500.csv Complete · 11d ago	0.33396	0.33678	☐
submission_DL_tfidf500.csv Complete · 11d ago · DL tiidf with 48 validation	0.26942	0.27034	☐
submission_decision_tree.csv Complete · 11d ago · template use wrong class name	0.26942	0.27034	☐
submission_decision_tree.csv Error · 11d ago · Using Decision Tree			
submission (1).csv Complete · 12d ago	0.31219	0.31627	☐

Model Training

Abstract

In the Kaggle competition, I tried several methods. For embedding, I primarily used TF-IDF, though I also attempted using Word2Vec, but the process took too much time. Regarding model selection, I experimented with three machine learning models: Random Forest, Decision Tree, and XGBoost. Additionally, I also tried using deep learning methods.

Interpret Data

- **Metadata Fields**:
  - `_score`: unknown
  - `_index` : unknown
  - `_crawldate`: date when data being crawed
  - `_type`: likely for classification purposes.
- **Tweet Data (`_source.tweet`)**:
  - `hashtags`: A list of hashtags present in the tweet.
  - `tweet_id`: A unique identifier for the tweet.
  - `text`: The content of the tweet

Data Preprocessing

technique

- **Text Cleaning**:
  - Remove punctuation, numbers, special characters, and extra spaces.
  - Convert text to lowercase.
- **Tokenization**:
  - Break text into words or sentences using libraries like `NLTK`, `spaCy`, or `transformers`.
- **Lemmatization/Stemming**:
  - Reduce words to their root form (e.g., "running" → "run").

Prepare data for training model

Due to the large size of the dataset, I chose to sample a certain percentage of the data for training. Even so, using just 20% of the data still required about an hour of training. Once I found suitable parameter combinations, I used the entire dataset for training. While this approach is not entirely rigorous, it serves as a useful reference.

Machine Learning

Training model & Accuracy

For the machine learning model selection, I tried XGBoost, Random Forest, and Decision Tree. Among them, XGBoost performed very poorly, producing outputs limited to only three emotion categories, resulting in poor outcomes. Initially, I determined that this was not a data issue, as the input was consistent across all models. Therefore, I decided not to use XGBoost as my model. As for Decision Tree and Random Forest, while Decision Tree trained faster, Random Forest produced better results. Ultimately, I chose to fine-tune Random Forest to find the best-performing model.

Try different combination of parameters

I tried a total of five parameters for Random Forest, each with three different values, resulting in 3^5 combinations. Ultimately, I found that the combination of max_depth=20 and max_features="sqrt" performed the best. However, the

accuracy was around 0.44 for all combinations. The primary goal was to find a suitable set of parameters to avoid overfitting caused by a lack of parameter control.

Deep Learning

I also tried using deep learning methods to train the model. Although the results were better than Random Forest, the training time was significantly longer. Therefore, I decided to first focus on fine-tuning the Random Forest model before adjusting the deep learning approach.

1. Model Definition

- **Input Layer**: Defines the input dimension
- **Hidden Layers**:
  - **First Hidden Layer: Fully connected layer with 64 neurons.**
  - Second Hidden Layer: Another dense layer with 64 neurons and ReLU activation.
- **Output Layer**:
  - Activation function: Softmax to output probabilities for multi-class classification.
- Model Compilation:
  - **Optimizer**: Adam - An adaptive learning rate optimization algorithm.
  - **Loss Function**: Categorical crossentropy - Suitable for multi-class classification.
  - **Metrics**: Accuracy to track the model's performance during training.

2. Training the Model

The training process involves feeding the model data, adjusting weights based on errors (loss), and tracking metrics over epochs.

3. Visualization of Training Performance

- Accuracy Plot
  - **Training Accuracy (`accuracy`)**: Indicates how well the model performs on the training data.

    - **Validation Accuracy (`val_accuracy`)**: Indicates how well the model generalizes to unseen data.
- Loss Plot:
    - **Training Loss (`loss`)**: Measures the error on the training data.
    - **Validation Loss (`val_loss`)**: Measures the error on the validation data.
- Insights from the Plots:
    - **Accuracy**:
        - If training and validation accuracy increase and converge, the model is learning well.
        - If training accuracy increases while validation accuracy stagnates or decreases, overfitting may occur.
    - **Loss**:
        - If training and validation loss decrease and converge, the model is optimizing effectively.
        - If validation loss increases while training loss decreases, overfitting is likely.

Use OpenAI embedding

I originally planned to use OpenAI's API for embedding, but I found that it required a lot of time—processing 300K records would take over 3 hours. Therefore, I determined that this method was not suitable for my current situation.

Conclusion

This competition provided valuable insights into text preprocessing, model selection, and the importance of hyperparameter tuning. I experimented with a variety of methods, including traditional machine learning, deep learning, and embedding techniques, which enriched my understanding of their trade-offs and applications. However, hardware limitations and the time required for processing large datasets significantly shaped my decisions and constrained my exploration of more advanced methods. This experience reinforced the importance of balancing resource availability with methodological rigor and inspired me to seek more efficient approaches in future projects.