Github issue auto labeling model - Use Facebook/react as example

1. Introduction

This project applies machine learning to automate the classification and labeling of GitHub issues in the Facebook/react repository. Automating this process can significantly enhance issue management efficiency, leading to faster resolution times and better project management. The project involves data preprocessing, model training using various machine learning techniques, and an evaluation of these methods' effectiveness.

2. Data Mining & Preprocessing

Step 1: Parse data from Github issue

Using Github API to get issue data from Facebook/react repository. Collecting issue title, body, labels, comments, and other information. Collecting 5 types of labels: Bug, Discussion, Enhancement, Feature Request and Question.

Step 2: Remove template from description

Remove template from issue description. The template is a markdown file that contains the following information:

```
keys_to_remove = [
       "### Website or app",
       "### Repro steps",
       "### How often does this bug happen?",
       "### DevTools package (automated)",
       "### DevTools version (automated)",
       "### Error message (automated)",
       "### Error call stack (automated)",
       "### Error component stack (automated)",
       "### GitHub query string (automated)",
       "No response",
       "## Steps To Reproduce",
       "## The current behavior",
       "## The expected behavior",
   ]
```

Replace \n and \r with white space in the description

Step 3. Seperate code and non-code part

Seperate code and non-code part in issue description. The code part is surrounded by `and` . The non-code part is the rest of the description.

Step 4. Remove noise from non-code part

Remove Markdown link syntax, URLs and HTML tags from description.

Step 5. NLP preprocessing

Remove stop words. Lemmatize the tokens.

The decision to use `stopwords_removal_lemmatization` instead of the original description, simple stopwords removal, or the lemmatization was made to reducing noise. Lemmatization was preferred over stemming for its ability to reduce words to their base or dictionary form while maintaining their contextual meanings. The removal of stopwords further helped in focusing on the more meaningful words in the reviews.

Step 6. Remove duplicate issues (only for multi-class classification)

Remove duplicate issues. The duplicate issues are the issues that have the same issue_id, title and description.

3. Model Training and Evaluation

Number of Data:

Multi-class classification:

Type: Bug: 1129 Type: Question: 326 Type: Feature Request: 327 Type: Enhancement: 195 Type: Discussion: 344

Multi-label classification:

Type: Bug: 1135 Type: Question: 328 Type: Feature Request: 336 Type: Enhancement: 199 Type: Discussion: 355

Multi-class classification

- Use Random Forest, and Gradient Boosting as the classification model.
- Feature Engineering:
    - Use TF-IDF to vectorize the description. Use the vectorized description as the feature.
    - Use lstm (Long Short-Term Memory) to vectorize the description. Use the vectorized description as the feature.
    - Use word2vec to vectorize the description. Use the vectorized description as the feature.

- For Data Imbalance:
  - Use SMOTE to balance the data.(Synthetic samples rather than duplicating existing ones.)
  - Use class_weight to balance the data.

Algorithm	Setting	Bug F1 Score	Discussion F1 Score	Enhancement F1 Score	Feature Request F1 Score	Question F1 Score	Overall Accuracy / F1 Score
Random Forest	tf-idf + title/lemmatization + human words/lemmatization	0.77	0.36	0.05	0.43	0.44	0.61 / 0.55
Random Forest	tf-idf + title/lemmatization + human words/lemmatization + SMOTE	0.79	0.34	0.31	0.51	0.42	0.61 / 0.59
Random Forest	tf-idf + title/lemmatization + human words/lemmatization + SMOTE + keywords	0.89	0.64	0.63	0.66	0.60	0.76 / 0.76
Gradient Boosting	tf-idf + title/lemmatization + human words/lemmatization	0.80	0.44	0.29	0.52	0.32	0.62 / 0.60
Gradient Boosting	tf-idf + title/lemmatization + human words/lemmatization + SMOTE	0.81	0.42	0.29	0.55	0.42	0.62 / 0.62
Gradient Boosting	tf-idf + title/lemmatization + human words/lemmatization + SMOTE + keywords	0.92	0.71	0.61	0.74	0.61	0.78 / 0.79
Random Forest	word2Vec + title/lemmatization + human words/lemmatization + SMOTE	0.72	0.31	0.23	0.26	0.20	0.47 / 0.48
LSTM	title/lemmatization + human words/lemmatization + SMOTE	0.76	0.23	0.32	0.35	0.23	0.50 / 0.51
LSTM	title/lemmatization + human words/lemmatization + SMOTE + keywords	0.90	0.37	0.34	0.37	0.37	0.64 / 0.65

Multi-label classification

- Use Random Forest, and Gradient Boosting as the classification model.
- Feature Engineering:
  - Use TF-IDF to vectorize the description. Use the vectorized description as the feature.
- For Data Imbalance:
  - Use RandomOverSampler to balance the data.(RandomOverSampler is a way of dealing with imbalanced data in which the minority class is over-sampled by randomly replicating some of the observations.)

Algorithm	Setting	Bug F1 Score	Discussion F1 Score	Enhancement F1 Score	Feature Request F1 Score	Question F1 Score	Overall Accuracy / F1 Score	Hamming Loss
Random Forest	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization	0.82	0.16	0.05	0.09	0.18	0.43 / 0.47	0.13
Random Forest	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization + ROS	0.77	0.21	0.09	0.09	0.18	0.38 / 0.46	0.14
Random Forest	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization + KW	0.93	0.44	0.13	0.40	0.40	0.57 / 0.66	0.09
Random Forest	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization + KW + ROS	0.91	0.49	0.32	0.53	0.40	0.57 / 0.68	0.09
Gradient Boosting	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization	0.81	0.33	0.19	0.34	0.31	0.49 / 0.56	0.12
Gradient Boosting	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization + ROS	0.73	0.38	0.23	0.30	0.25	0.38 / 0.51	0.14
Gradient Boosting	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization + KW	0.92	0.67	0.56	0.65	0.56	0.68 / 0.77	0.08
Gradient Boosting	tf-idf + title_stopwords_removal_lemmatization + human_words_stopwords_removal_lemmatization + KW + ROS	0.87	0.67	0.59	0.73	0.53	0.63 / 0.75	0.08

4. Analysis and Insights

Keywords:

The use of `gensim` for keyword extraction and incorporating these keywords into the tf-idf vectorization process has shown significant improvements in both accuracy and F1 score. Keywords likely provide a clearer signal for the model, helping it distinguish between various categories.

Data Imbalance:

The dataset exhibits a notable imbalance, particularly with the 'Bug' label being more prevalent. This imbalance can skew the model's performance, making it more adept at identifying 'Bug' issues while struggling with less frequent labels.

SMOTE generates synthetic samples, which can be beneficial in providing varied examples to the model. However, it might also introduce synthetic noise.

RandomOverSampler simply duplicates existing minority instances, which can lead to overfitting, especially in smaller datasets.

Overfitting Concerns with Oversampling:

While oversampling techniques improve model performance on minority classes, they come with the risk of overfitting. This is particularly true for RandomOverSampler, as it increases the frequency of minority examples without introducing new information. Consequently, the model might become too tailored to the training data, hindering its ability to generalize to new, unseen data.

Model Selection and Performance:

The use of different models (Random Forest, Gradient Boosting, LSTM) with various settings shows varied effectiveness. For instance, Gradient Boosting with tf-idf vectorization and keyword inclusion appears to significantly enhance performance across almost all labels.

Future Directions:

Further research could explore more advanced natural language processing techniques, like BERT or GPT, which might capture the context of issues more effectively than traditional methods.

Analysis of LSTM and Word2Vec Performance:

GitHub issues often contain a mix of natural language and technical terms or code snippets. This combination can make it challenging for LSTM and Word2Vec to capture the context accurately, especially if the training data is not sufficiently large or diverse to encapsulate the variety of language used in these issues.

LSTM models, being a type of recurrent neural network, are particularly sensitive to the nuances of sequential data. They require a substantial amount of data to train effectively. If the LSTM model was trained on a limited dataset, or if the hyperparameters were not optimized, this could lead to suboptimal performance. Similarly, the effectiveness of Word2Vec depends on the quality of the vector representations it generates, which again hinges on the quantity and diversity of the training data.

Challenges of Multi-label vs. Multi-class Classification:

Multi-label datasets often suffer from label imbalance, where some labels appear much more frequently than others. This imbalance can skew the model's learning, making it biased towards more common labels and less accurate in predicting rare labels.

5. Conclusion:

This study highlights the successful application of machine learning in automating the labeling of GitHub issues, with a focus on the Facebook/react repository. Key takeaways include the benefit of keyword extraction for improved accuracy and the need to address data imbalance. Ensemble methods like Gradient Boosting have shown promising results in classification tasks. Future efforts should concentrate on incorporating advanced NLP techniques and adapting the model to evolving data.