Assignment 2

Fabio Zanotti, Antonio Morelli, Federico Rullo and **Edoardo Conca**
Master's Degree in Artificial Intelligence, University of Bologna
{ fabio.zanotti, antonio.morelli, federico.rullo, edoardo.conca }@studio.unibo.it

Abstract

The experiment aimed to develop and evaluate models for a multi-label classification task on a dataset containing samples labeled with multiple categories. Three different model architectures were explored: CClassifier, CPClassifier, and CPSClassifier. Each model was trained and evaluated using per-category and macro F1-scores as performance metrics. Additionally, precision-recall curves were plotted to analyze the trade-off between precision and recall. The results showed that the CPSClassifier consistently outperformed the other models, demonstrating the effectiveness of incorporating additional input features for improved classification performance.

1 Introduction

Multi-label classification tasks involve predicting multiple labels for each sample in a dataset. These tasks are prevalent in various domains, including natural language processing and healthcare. In this experiment, we developed and evaluated models for a multi-label classification task using a dataset containing samples labeled with psychological constructs. The objective was to predict these psychological construct labels based on text inputs. We explored three different model architectures: CClassifier, CPClassifier, and CPSClassifier, each incorporating different combinations of input features, based on BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art model in natural language processing, which serves as the backbone for our classification architectures due to its robust contextual understanding capabilities.The experiment aimed to assess the performance of these models using various evaluation metrics, including per-category and macro F1-scores, as well as precision-recall curves and a final comparison with a Random Uniform Classifier and a Majority Classifier.

2 System description

The models evaluated in this study include:

- BERT-based models - these models leverage pre-trained BERT architectures fine-tuned on our dataset.
- Random Uniform Classifier - a baseline model that predicts classes uniformly at random.
- Majority Classifier: a baseline model that always predicts the most represented class in the dataset.

In the following list, the evaluated BERT model variants are described more precisely:

- ◇ CClassifier: Basic BERT model for classification with conclusion text inputs 1.
- ◇ CPClassifier: BERT model incorporating both premise and conclusion text inputs 2.
- ◇ CPSClassifier: BERT model that extends the previous models by incorporating premise, conclusion, and stance text inputs 3.

We selected the tiny version of the BERT model, specifically the prajjwal1/bert-tiny, due to its lightweight nature, suitable for resource-constrained environments. We employ the AutoTokenizer to automatically load the tokenizer associated with the chosen model. Finally, we load the datasets using the DataLoader class, iterating through the dataset in batches of 16, with data shuffling enabled to ensure variability in each epoch. From HuggingFace, we acknowledge that the maximum sequence length for our model is 512 tokens. The stance input is encoded into a numerical format, and the same model instance is used to encode both premise and conclusion inputs. The models are then trained on the training dataset and evaluated on the validation set. Weighted loss functions are used to address class imbalance.

3 Experimental setup and results

3.1 Parameter Tuning and Metrics

To optimize model performance, we experimented with various learning rates (1e-3, 1e-5, 2e-5), batch sizes (8, 16, 32), training epochs (5, 10, 15), and weight decay values to prevent overfitting. Each parameter combination was evaluated on the validation set to find the optimal configuration.
Model performance was assessed using sklearn's f1_score function. The average parameter allowed us to compute per-category binary F1-scores, while the macro parameter provided unweighted means across all labels, disregarding class imbalance.
Our data analysis revealed significant class imbalance. To address this, we calculated class weights based on the inverse frequency of each class, ensuring that under-represented classes received higher weights. This encouraged the model to focus more effectively on these classes by emphasizing their error during training.

3.2 Results

The experimental results highlight the critical role of incorporating additional input features, such as premise, conclusion, and stance, in improving model performance for multi-label classification tasks. The CPSClassifier, which integrates all three input features, demonstrated superior performance and emerged as the best-performing model on both validation and test sets.

Table 1: F1 Scores on Validation Set

	Macro F1 Scores
CClassifier	0.679
CPClassifier	0.73
CPSClassifier	**0.731**

Table 2: F1 Scores on Test Set

	Macro F1 Scores
CClassifier	0.610
CPClassifier	0.692
CPSClassifier	**0.698**
Random Uniform Classifier	0.503
Majority Classifier	0.431

Table 3: F1 Scores on Test Set - CPSClassifier

	Macro F1 Scores
Openness to change	0.53
Self enhancement	0.545
Conservation	0.829
Self transcendence	0.888
Average	**0.698**

4 Discussion

The similar performances between the CPSClassifier and the CPClassifier suggest that the slight improvement might be due to chance, confirming that stance values, being equally distributed, do not significantly enhance performance.
Moreover, correctly predicting "Openness to Change" and "Self-Enhancement," which are less represented classes, contributes highly to the overall score differences. This indicates that our models handle class imbalance effectively.
In baseline comparisons, the BERT model consistently outperforms both the random and majority classifiers in precision, recall, and F1-score.This highlights the value of considering multiple contextual factors in text classification.
Analyzing the precision-recall curves provided insights into the models' trade-offs between precision and recall, emphasizing the challenges in balancing true positives and false positives. These curves illustrated how the CPSClassifier maintained a better balance compared to other models, showcasing its robustness in handling imbalanced datasets. While the CClassifier generally performs worse than the other two variants, tending to output similar predictions. When highly represented classes are true, the models rarely fail to predict correctly, underscoring the imbalance between positive and negative samples.

5 Conclusion

In conclusion, the experiment successfully developed and evaluated models for a multi-label classification task on a dataset containing samples labeled with psychological constructs. The CPSClassifier emerged as the best-performing model, demonstrating the importance of incorporating additional input features for improved classification performance. The experiment's findings suggest that fine-tuning model architectures and input features can significantly improve classification performance and offer deeper insights into the underlying data distribution and model behavior. The integration of various textual elements allows for a more comprehensive understanding and accurate classification, which is crucial for tasks that require detailed contextual analysis.

6 Links to external resources

Assignment_2

References

Johannes Kiesel, Milad Alshomary, Nicolas Handke, Xiaoni Cai, Henning Wachsmuth, and Benno Stein. 2022. Identifying the human values behind arguments. Dublin, Ireland. Association for Computational Linguistics.

James McCaffrey. 2020. Multi-class classification using pytorch: Defining a network.

Appendix

Figures

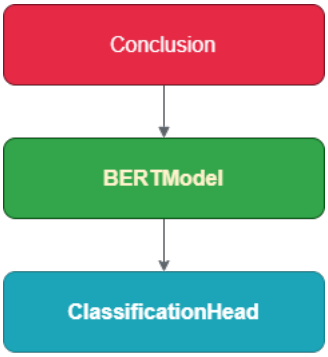

Figure 1: CClassifier architecture

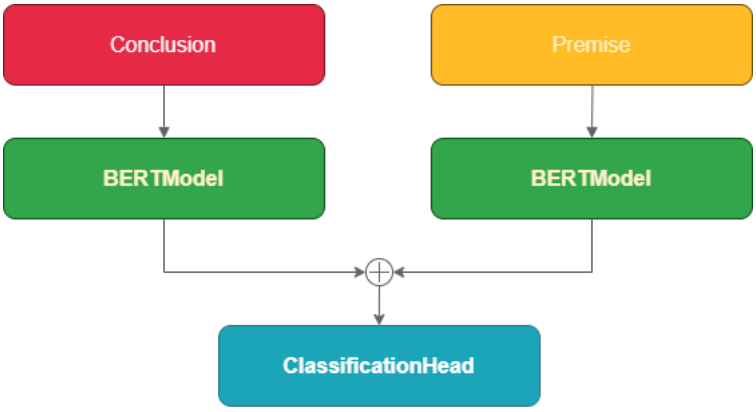

Figure 2: CPClassifier architecture

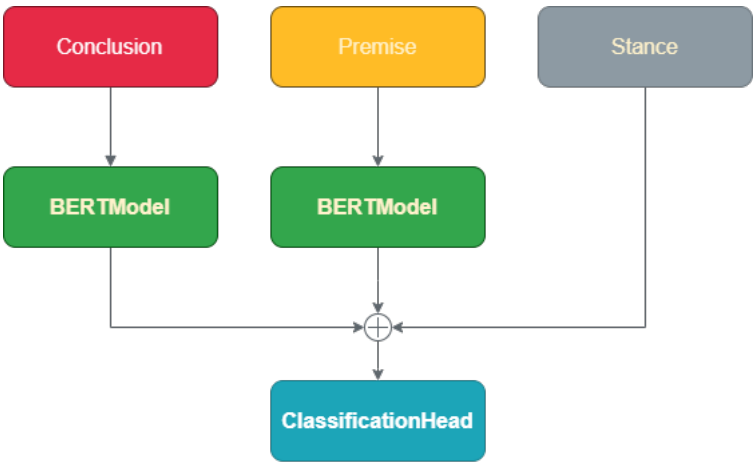

Figure 3: CPSClassifier architecture