Sabanci University 2024 – 2025 Fall Term

CS445 - Natural Language Processing

Project Individual Report

1. Introduction

Our project focused on building an intent detection system using the **CLINC150** dataset. This dataset is widely recognized for benchmarking intent classification tasks, as it contains **22,500** utterances across **150 distinct intent classes**, with predefined train, validation, and test splits. The diverse nature of the dataset makes it highly relevant for real-world applications such as virtual assistants and customer support bots. Our final goal was to design a robust **NLP pipeline** including data preprocessing, model training, and evaluation that could achieve **high accuracy** and **F1 scores**. By doing so, we aimed not only to demonstrate the efficiency of various modeling techniques but also to gain deeper insights into the best practices for intent detection in contemporary NLP systems.

2. Unique Contributions and Implementation Details

After our team completed **data preprocessing** and conducted initial **feature extraction** via TF-IDF, we experimented with various classic machine learning algorithms, including **Naive Bayes**, **Support Vector Machines (SVM)**, and simple **Neural Networks** (Perceptrons). Through careful **hyperparameter tuning**, we achieved approximately **83% accuracy** on the test set. Despite this success, we recognized room for improvement and began exploring more advanced **embeddings** approaches.

2.1 FastText Embeddings

To move beyond TF-IDF, I introduced **FastText** for dense vector embeddings. Following the official FastText documentation, I preprocessed the data in the format required for building FastText embeddings and ran various hyperparameter experiments. Specifically, I fine-tuned parameters such as learning rate, n-gram length, loss function, and number of epochs. The best configuration : **lr=1.0**, **epochs=50**, and **loss=softmax** resulted in an accuracy of around **89% on the validation set**. Subsequently, retraining on both the training and validation sets yielded a test accuracy of **89.72%**, which significantly outperformed our initial TF-IDF-based models.

2.2 Contextual Embeddings with RoBERTa-base

While FastText offered a notable performance boost, we still fell short of the highest reported **accuracy** and **F1** scores for the CLINC150 dataset. Recognizing that **transformer-based models** often outperform other methods in NLP tasks, I proposed **fine-tuning a RoBERTa-base model** with a simple **linear classifier** on top. This architecture is a well-established approach for using pretrained language models in classification tasks.

1. **Hyperparameter Testing**: I initially conducted short training runs (3 epochs) to identify effective hyperparameters, finding that an **AdamW** optimizer with a **1e-5** learning rate offered a favorable balance of convergence and generalization.
2. **Extended Training**: Building on these findings, I trained the RoBERTa-base + linear classifier setup for **40 epochs**, again using **AdamW** and a **1e-5** learning rate. Throughout training, I monitored validation performance to adjust training steps as needed.
3. **Hardware and Runtime**: The model was trained on **Kaggle's P100 GPUs**, which took approximately **6 hours** to complete the entire 40-epoch run.
4. **Final Results**: This approach yielded **>96% accuracy** on the test set—an excellent outcome, especially considering the model was customized and trained from scratch for this specific dataset.

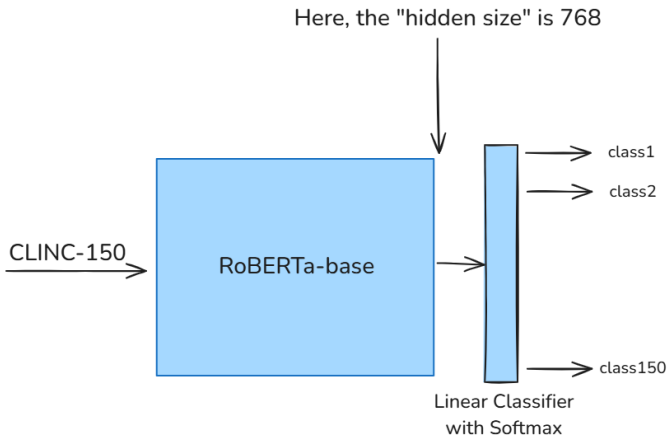

Figure 1: RoBERTa-base fine-tuned for CLINC150

3. Challenges

For the **FastText** approach, the primary challenge involved studying the official documentation for supervised learning and learning how to correctly format the dataset. Additionally, hyperparameter tuning required multiple training runs on the training set to optimize parameters like n-gram length, learning rate, and loss function, before finalizing a model that reached **~90% accuracy** on the test set.

The most challening aspect of my contribution was **fine-tuning RoBERTa-base**. Although various tutorials were available online, this was my first time fine-tuning a BERT-like model. It required understanding how to use the **Transformers** library in Python and adapting its usage specifically for the **CLINC150 dataset**, including data loading, model configuration, and training loop design.

4. Conclusion

Overall, this project offered valuable insights into **intent detection** using the CLINC150 dataset. We began with traditional **TF-IDF** feature extraction and machine learning algorithms, achieving around **83% accuracy**. By shifting to **FastText** embeddings, I demonstrated how careful hyperparameter tuning could push performance to nearly **90%** on

the test set. Finally, **fine-tuning a RoBERTa-base model** allowed us to surpass **96% accuracy**, reflecting the powerful benefits of **transformer-based** architectures for NLP tasks.

Through these steps, I gained significant experience in tackling data preparation, hyperparameter optimization, and the complexity of **transformer model fine-tuning**.

5. References

Hugging Face. (2019). *Fine-tune a pretrained model*. Retrieved January 5, 2025, from https://huggingface.co/docs/transformers/en/training

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). *Bag of tricks for efficient text classification*. arXiv. https://arxiv.org/abs/1607.01759

Lin, Y.-T., Papangelis, A., Kim, S., Lee, S., Hazarika, D., Namazifar, M., Jin, D., Liu, Y., & Hakkani-Tur, D. (2023). *Selective in-context data augmentation for intent detection using pointwise V-information*. arXiv. https://arxiv.org/pdf/2302.05096v1

Mishra, P. (2022). *Fine-tune BERT for text classification*. Kaggle. Retrieved January 5, 2025, from https://www.kaggle.com/code/pritishmishra/fine-tune-bert-for-text-classification?scriptVersionId=116951029