Intent Detection Project - Final Report

1. Introduction

This project focuses on intent detection, a crucial task in natural language processing (NLP) that involves classifying user utterances into predefined intent categories. Intent detection is widely used in applications such as dialogue systems, virtual assistants, and search engines (Ma et al., 2022). However, the variability of intents and the constant introduction of new ones make achieving high accuracy challenging.

To address these challenges, we implemented a Naive Bayes model as a baseline, achieving a test accuracy of 56%. For more advanced approaches, we fine-tuned DistilBERT (Zhang et al., 2021) with a Softmax-based classifier and experimented with Pointwise V-Information (PVI) filtering as a data preprocessing technique. The DistilBERT+Softmax model without PVI filtering achieved a test accuracy of 66.50%, while the model with PVI filtering improved this significantly to 93.95%. This improvement highlights the importance of high-quality data preprocessing. Key parameters, such as learning rate, dropout rate, and epochs, were fine-tuned during the experiments. Metrics such as validation loss, test accuracy, and F1 scores were used for evaluation.

In this report, we detail the methodologies, results, and insights gained from our journey in developing and refining intent detection models.

2. Methodology

- **Dataset Selection**

For this project, we initially considered two datasets:

1. *Banking77 (Casanueva et al., 2020)*: A dataset containing 13,083 user queries from the banking domain labeled across 77 intent classes. We split the data into 80% training, 10% validation, and 10% testing. Its complexity and diversity made it a suitable choice for real-world applications.
2. *Intent Classification for IDE Functionalities (Usmani, 2023)*: A simpler dataset focusing on intent classification for integrated development environment (IDE) functionalities. Although it offered a focused application, its simplicity rendered it less suitable for evaluating advanced models.

Ultimately, we chose the Banking77 dataset due to its robustness and complexity, which provided a challenging environment for evaluating our methodologies.

Model Implementation

To implement PVI filtering using the distilBERT model effectively, we adopted a structured process that combines fine-tuning, scoring, and filtering. First, the Banking77 dataset, consisting of 13,083 labeled instances, was selected as the base dataset. To simulate noise, 5,000 noisy instances were generated using the GPT4o model and concatenated with the original dataset, resulting in a mixed dataset of 18,083 instances. We chose DistilBERT for its computational efficiency and robust performance and fine-tuned it on the clean portion of the Banking77 dataset. During fine-tuning, hyperparameters such as a learning rate of 2e-5, batch size of 16, and up to 5 epochs were used. Cross-entropy loss was minimized on the training set, and model performance was validated on a held-out validation set.

After fine-tuning, PVI scores were calculated for each instance in the mixed dataset. The score for an instance was derived using the following formula:

$PVI(x \rightarrow y) = -\log_2 g^*[\varnothing][y] + \log_2 g'[x][y].$ (Lin et al., 2024)

Here, $g^*[\varnothing][y]$ represents the probability assigned to label y when the model processes an empty string and $g'[x][y]$ is the probability assigned to y when the model processes the text instance x. Both probabilities were obtained by passing the respective inputs through the fine-tuned distilBERT model and extracting the softmax outputs. Base-2 logarithms were applied to compute the final score of the given instance.

To determine the threshold for filtering, the mean (μ) and standard deviation (σ) of the PVI scores across the mixed dataset were calculated. The threshold was set to 4.5 ($T=\mu-2\sigma$), eliminating low-scoring instances likely to be noisy. Instances with PVI scores above this threshold were retained, while those below were removed. Following this process, 12,903 instances were classified as valid, including 19 noisy instances misclassified as valid and 197 clean instances from the original dataset misclassified as noisy.

We explored three approaches:

1. **Naive Bayes Classifier:** This served as a baseline. We used Term Frequency-Inverse Document Frequency (Tf-idf) features and applied standard supervised learning techniques. It achieved a test accuracy of 56%.
2. **DistilBERT + Softmax (without PVI Filtering)**: Leveraging the pre-trained DistilBERT model, we added a Softmax-based classifier for intent detection. Fine-tuning was performed using the Banking77 dataset with hyperparameters: learning rate = 5e-05, dropout = 0.1, epochs = 5, and batch size = 16. This approach achieved a test accuracy of 66.50%.
3. **DistilBERT + Softmax (with PVI Filtering):** To enhance performance, we implemented PVI filtering to eliminate noisy data. PVI scores were computed for each instance, and low-quality samples were removed. Augmentation techniques created additional samples to balance classes. The model's test accuracy improved significantly to 93.95%.

Challenges during implementation included resource constraints and difficulties in generating quality augmented data. Despite these obstacles, PVI filtering proved to be an effective preprocessing method for improving dataset quality.

Training and Evaluation

Data splits for the Banking77 dataset were 80% training, 10% validation, and 10% testing. Key evaluation metrics included accuracy, F1 scores, precision-recall curves, and validation/test loss. Confusion matrices and precision-recall curves visually demonstrated the impact of PVI filtering on classification performance.

3. Results

Naive Bayes Classifier

- Validation Accuracy: 56%
- Validation F1-score: 56%
- Test Accuracy: 56%
- Test F1-score: 56%

DistilBERT + Softmax (without PVI Filtering)

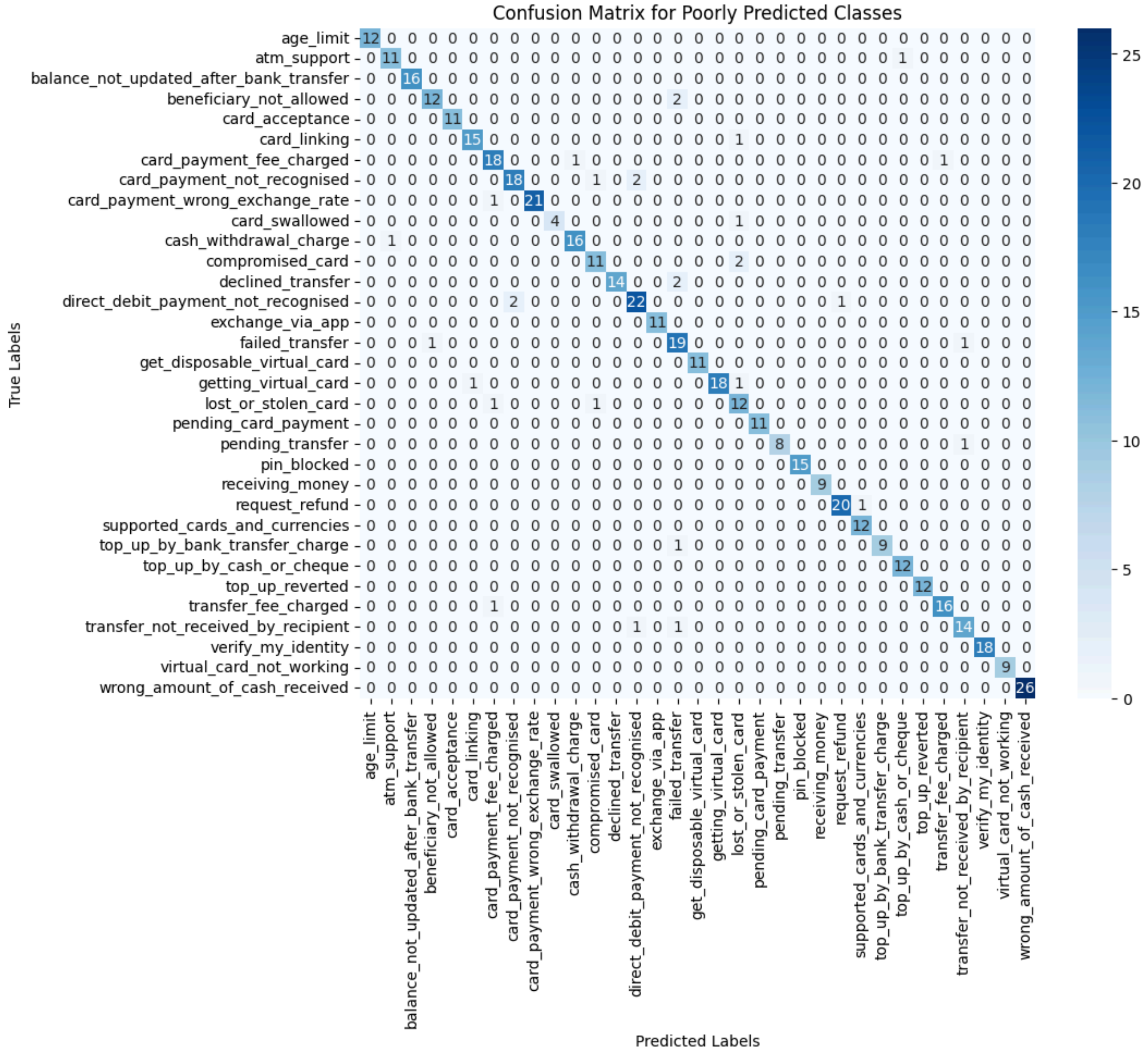
- Validation Accuracy: 67.81%
- Validation F1-score: 68.87%
- Test Accuracy: 66.50%
- Test F1-score: 67.43%

DistilBERT + Softmax (with PVI Filtering)

- Validation Accuracy: 94.10%
- Validation F1-score: 94.10%
- Test Accuracy: 93.95%
- Test F1-score: 93.97%

Confusion matrices and precision-recall curves confirmed that PVI filtering significantly reduced noise and improved both precision and recall.

Classes with prediction accuracy below 93.0%



	age_limit	atm_support	balance_not_updated_after_bank_transfer	beneficiary_not_allowed	card_acceptance	card_linking	card_payment_fee_charged	card_payment_not_recognised	card_payment_wrong_exchange_rate	card_swallowed	cash_withdrawal_charge	compromised_card	declined_transfer	direct_debit_payment_not_recognised	exchange_via_app	failed_transfer	get_disposable_virtual_card	getting_virtual_card	lost_or_stolen_card	pending_card_payment	pending_transfer	pin_blocked	receiving_money	request_refund	supported_cards_and_currencies	top_up_by_bank_transfer_charge	top_up_by_cash_or_cheque	top_up_reverted	transfer_fee_charged	transfer_not_received_by_recipient	verify_my_identity	virtual_card_not_working	wrong_amount_of_cash_received
age_limit	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
atm_support	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
balance_not_updated_after_bank_transfer	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
beneficiary_not_allowed	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
card_acceptance	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
card_linking	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
card_payment_fee_charged	0	0	0	0	0	0	18	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
card_payment_not_recognised	0	0	0	0	0	0	0	18	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
card_payment_wrong_exchange_rate	0	0	0	0	0	0	1	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
card_swallowed	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cash_withdrawal_charge	0	1	0	0	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
compromised_card	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
declined_transfer	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
direct_debit_payment_not_recognised	0	0	0	0	0	0	0	2	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
exchange_via_app	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
failed_transfer	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
get_disposable_virtual_card	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
getting_virtual_card	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	18	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
lost_or_stolen_card	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pending_card_payment	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0
pending_transfer	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	1	0	0	0
pin_blocked	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0
receiving_money	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0
request_refund	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1	0	0	0	0	0	0	0	0
supported_cards_and_currencies	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0
top_up_by_bank_transfer_charge	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
top_up_by_cash_or_cheque	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0
top_up_reverted	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0
transfer_fee_charged	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	0	0	0	0
transfer_not_received_by_recipient	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0
verify_my_identity	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0
virtual_card_not_working	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0
wrong_amount_of_cash_received	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26

Additional Methods Explored

During the project, we tested several alternative approaches:

1. **TinyBERT:** Although initially considered for its lightweight architecture, TinyBERT's performance was suboptimal, with extended runtimes due to being run on a CPU instead of a GPU. DistilBERT emerged as a superior alternative.
2. **Activation Functions:** Various activation functions, including ReLU, LeakyReLU, and Sigmoid, were tested. ReLU consistently delivered the best results, improving accuracy and F1 scores.
3. **Data Augmentation:** Several data augmentation techniques were explored to enhance model generalization and robustness:

- **Back-translation:** Translating sentences to another language and back to generate new variations.
- **Random swaps:** Swapping words randomly within sentences.
- **Random deletions:** Removing random words to simulate noise.

4. **Self-Pooling:** Self-pooling layers were tested as an alternative to traditional pooling methods.

These experiments, though not all successful, provided valuable insights into the complexities of intent detection and the trade-offs between model complexity and performance.

4. Discussion

The selection of the Banking77 dataset was instrumental in evaluating our methodologies due to its diverse and complex intents. However, its challenges highlighted the importance of robust preprocessing techniques like PVI filtering. PVI filtering offers notable advantages: it eliminates noisy data and enables the model to better handle such data, leading to improved performance. However, it also has certain disadvantages: it may inadvertently remove some clean data, and since data augmentation was not integrated into our specific PVI filtering implementation, we had to rely heavily on most of the clean data to fine-tune the PVI filtering model. Incorporating data augmentation could address this limitation by enabling the model to fine-tune itself effectively even with limited dataset sizes.

Our approach, leveraging DistilBERT with a Softmax-based classifier, demonstrated clear advantages in accuracy and efficiency. The addition of PVI filtering further amplified these benefits, underscoring the impact of high-quality data preprocessing. Compared to the Naive Bayes baseline and other existing systems, our approach achieved state-of-the-art performance for the selected task. However, limitations included resource constraints and difficulties in scaling augmentation techniques.

If we had more time and resources, we could explore using a Retrieval-Augmented Generation (RAG) system with related banking comments. Integrating a RAG system would enable the model to retrieve contextually relevant information, enriching its understanding of complex intents and improving classification accuracy. Additionally, we could investigate ensemble methods, using larger transformer models, or integrating semi-supervised learning techniques to handle unseen intents more effectively. These advancements would help address scalability challenges and enhance overall system robustness.

5. Conclusion

This project tackled the challenge of intent detection by developing and evaluating several models. Starting with a Naive Bayes baseline, we transitioned to advanced transformer-based models, ultimately achieving a test accuracy of 93.95% using DistilBERT with PVI filtering. The results highlight the critical role of data preprocessing and model selection in achieving high performance. Despite limitations, the project demonstrated the potential of combining advanced NLP models with targeted preprocessing to enhance intent classification systems.

References

- Hugging Face. (n.d.). Banking77 dataset. Hugging Face Datasets. Retrieved from https://huggingface.co/datasets/legacy-datasets/banking77
- Lin, Y.-T., Papangelis, A., Kim, S., Lee, S., Hazarika, D., Namazifar, M., Jin, D., Liu, Y., & Hakkani-Tur, D. (2024). Selective In-Context Data Augmentation for Intent Detection using Pointwise V-Information. *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL 2024)*.
- Usmani, M. A. (2023). Intent Classification for IDE Functionalities. Kaggle. https://www.kaggle.com/dsv/6122560. https://doi.org/10.34740/KAGGLE/DSV/6122560
- Zhang, H., Xu, H., Lin, T.-E., & Lyu, R. (2020). Discovering new intents with deep aligned clustering. Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2020).
- Hugging Face. (n.d.). DistilBERT: A distilled version of BERT. Hugging Face. Retrieved from https://huggingface.co/docs/transformers/en/model_doc/distilbert
- Zhang, J., Bui, T., Yoon, S., Chen, X., Liu, Z., Xia, C., Tran, Q. H., Chang, W., & Yu, P. (2021). Few-shot intent detection via contrastive pre-training and fine-tuning. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (pp. 1906–1912).