AUTOMATED INTENT CLASSIFICATION FOR BANKING CUSTOMERS

VAKATI, SAI VYSHALI vakati.s@northeastern.edu
SALIM, FATHIMA salim.f@northeastern.edu

Abstract

This research paper explores the optimization of Long Short-Term Memory (LSTM) networks for intent classification within the banking domain, employing the Banking77 dataset. The study examines various text preprocessing strategies—namely the inclusion and exclusion of punctuation, stop words, and lemmatization—and introduces out-of-vocabulary (OOV) handling and contractions. After rigorous experimentation, the optimal configuration excludes lemmatization but includes punctuation and stop words. Also, experimented with word2vec parameters like embedding dimension, window size and negative samples. Additionally, the paper delves into hyperparameter tuning and layer optimization within the LSTM and Bidirectional LSTM architecture, presenting a detailed analysis of their effects on the model's performance. The resulting system demonstrates substantial improvements in recognizing and classifying customer intents with precision.

1. Introduction and Background

In customer service, especially in banking, accurately classifying customer intents from text is key for improving service and efficiency. As artificial intelligence becomes more integrated into customer interactions, systems that can effectively interpret and respond to user queries are essential. Intent classification, a significant area of natural language processing (NLP), categorizes text into predefined groups and is crucial for routing customer inquiries.

The rise of deep learning has transformed NLP, with models like BERT and other transformer-based architectures setting new standards in tasks including intent classification. These models are highly effective due to their ability to understand deep contextual relationships in text, thanks to their complex network structures and self-attention mechanisms. However, these advanced models require substantial computational power and resources, which might not always be available, especially in real-time settings.

This highlights the value of Long Short-Term Memory (LSTM) networks, a type of recurrent neural network (RNN) excellent at processing sequential data like text. LSTMs are known for their capability to retain information over long periods, essential for understanding language context. Although they may not capture context as deeply as transformers, LSTMs offer a good balance between performance and computational efficiency, making them suitable for resource-limited applications.

This study uses the Banking77 dataset, which comprises a wide range of banking-related intents, to assess the effectiveness of LSTM networks in the banking sector. The dataset's complexity poses challenges that test the limits of LSTM's ability to handle detailed and diverse customer intents. Through systematic experiments, this paper examines various text preprocessing techniques and architectural optimizations to boost LSTM performance. By adapting the LSTM model to efficiently process and classify a wide array of banking queries, this research aims to show the practical viability of LSTMs as a competitive alternative to more resource-demanding models in certain NLP tasks. This work not only adds to the academic discussion on LSTM applications but also offers practical insights for practitioners implementing efficient NLP solutions in settings with limited resources.

2. Approach

This section explores the methods used to improve LSTM-based intent classification on the Banking77 dataset, including data preparation,

model design, and tuning. Our approach focuses on detailed preprocessing, careful handling of out-of-vocabulary (OOV) words, and thorough model tuning.

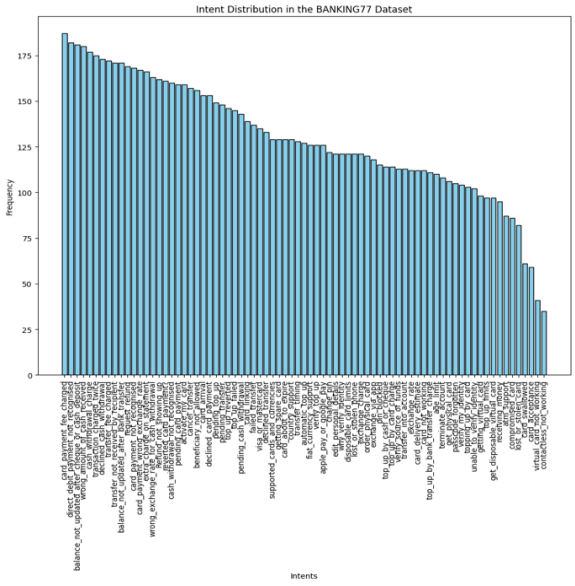


Figure 1 : Distribution of different classes

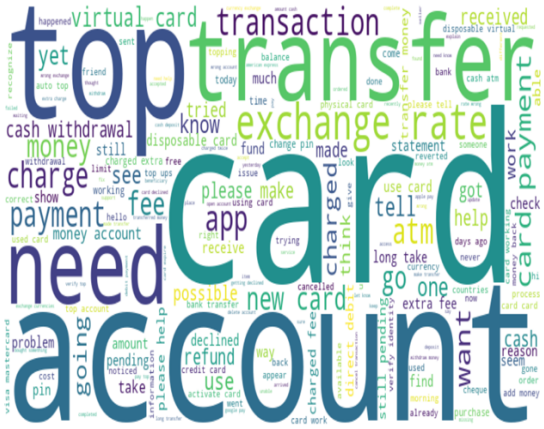


Figure 2: Most Frequent words

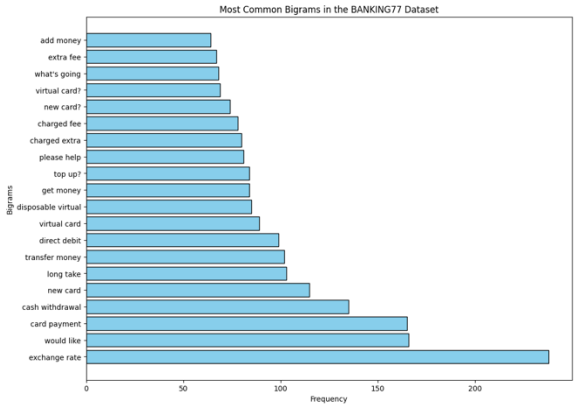


Figure 3 : Most common Bigrams

2.1 Data Preparation

Proper data preparation is key for the performance of NLP models. Our preprocessing steps were aimed at optimizing the text for LSTM networks:

a. **Tokenization and Vocabulary Building**: We started by breaking down the text into words or tokens. This is necessary to convert raw text into a suitable format.

b. **Handling of Punctuation and Stop Words**: Testing showed that keeping punctuation and stop words helped maintain important context needed for understanding customer intents. So, unlike typical text processing, we kept these elements.

c. **Omission of Lemmatization:** We did not use lemmatization, which usually simplifies words to their base form, because keeping words in their original form was important for recognizing differences between similar intents.

d. **Handling Out-of-Vocabulary Words:** Handling out-of-vocabulary (OOV) words, which are not in the pretrained

vocabulary but appear in the dataset, is a challenge. We included a special token <UNK> to represent OOV words and added to the vocabulary. We then treated the OOV words as zero vectors for their embeddings, which essentially means assigning them a placeholder that has no impact on the model's understanding of the text. Secondly, we allowed the model to learn the meaning of these OOV words from the surrounding context, which gives the model a chance to infer their meaning based on the words around them.

e. **Expansion of Contractions**: We expanded contractions like "can't" to "cannot" to reduce OOV words. This normalized the text and helped the model understand various expressions better.

f. **Word2Vec Embeddings**: We used Word2Vec with a CBOW model to create embeddings. This method predicts a word based on surrounding words, useful for grasping the banking terms where context matters a lot.

g. **Embedding Layer:** These embeddings were used in the LSTM's embedding layer to convert tokens into dense vectors. This layer captures deep meanings and relationships between words.

h. **Shuffling Batch Data:** Given the structured order of classes in the dataset, which could lead to biased model training outcomes, we implemented shuffling of batch data. This step was crucial to prevent the model from learning the sequence of the data instead of the underlying patterns required for effective intent classification. Shuffling ensures that each training batch contains a random example, thereby promoting robustness and reducing the likelihood of overfitting to a specific order of data presentation.

2.2 Model Architecture and Training

We designed the LSTM model to maximize its effectiveness:

Stacked LSTM Layers: We used several LSTM layers to deepen the learning. This helps the model understand everything from basic grammar to complex meanings.

Dropout: Dropout layers were used between the LSTM layers to avoid overfitting. This randomly turns off some neurons during training, helping the model generalize better.

Padding: Padding made all input data lengths uniform, ensuring consistent processing by the model.

2.3 Hyperparameter Tuning and Sampling Strategy:

We adjusted the hyperparameters to optimize the model's performance using random search.

```
Best hyperparameters:
units_lstm1: 128
dropout_1: 0.30000000000000004
units_lstm2: 64
dropout_2: 0.30000000000000004
units_dense: 128
dropout_3: 0.30000000000000004
initial_lr: 0.003982162357170975
```

Figure 4: Best hyperparameters after tuning

Optimization of LSTM Units: We tested different numbers of LSTM units to find the best balance for processing the data effectively.

Learning Rate and Batch Size: We fine-tuned the learning rate and batch size to ensure stable and efficient learning. The right learning rate helps the model minimize loss quickly, while a good batch size supports steady learning.

Dropout Rate: We experimented with various dropout rates to find the best one that prevents

overfitting while allowing the model to remain complex enough to learn effectively.

Word Embedding and Window Size Tuning: We experimented with various embedding dimensions and window sizes in our Word2Vec model. These parameters significantly influence the quality of the vector representations of words, which, in turn, affect the LSTM's ability to understand and classify text accurately. Embedding dimensions of 250 with window sizes ranging from 100 to 300, and Negative sampling values of 5, 10, and 20 to fine-tune the quality of the word embeddings. These adjustments were methodically assessed for their impact on the test accuracy, leading to an insightful understanding of the relationship between the Word2Vec hyperparameters and the LSTM model's classification performance.

Embedding Dimension	Window Size	Negative Samples	Test Accuracy (%)
250	200	20	77.82
250	200	10	76.46
250	100	20	76.40
250	200	5	75.91
200	200	20	75.13
250	300	20	74.97
250	300	10	74.84
250	100	10	74.45
250	100	5	73.41

Figure 5: Accuracies with different window sizes and sampling

2.4 Bidirectional LSTM Implementation

The BiLSTM layer was integrated into the existing LSTM stack, with careful adjustments to the input. and output configurations to accommodate the bidirectional flow of data. This layer was coupled with dropout layers to prevent overfitting and a dense output layer to classify the intents effectively. The learning rate and batch size were fine-tuned to be consistent with the bidirectional processing, ensuring that the model not only learns efficiently but also generalizes well across different data samples from the Banking77 dataset.

3. Results

This section presents the findings from extensive experiments conducted to optimize LSTM networks for intent classification using the Banking77 dataset. Various model configurations were assessed to determine their effectiveness in accurately classifying customer intents within the banking domain. Our exploration involved varying preprocessing strategies, LSTM architectures, and hyperparameter tuning to determine their impacts on model accuracy. The following summarizes the accuracies achieved under different configurations:

Model Configurations and Accuracies:

Table 3.1: Effect of Lemmatization and Text Preprocessing

Configuration	Accuracy
Without punctuation and stopwords, Lemmatized	75.62%
With punctuation and stop words, Lemmatized	81.69%

Table 3.2: Effect of Lemmatization Exclusion

Configuration	Accuracy
Without punctuation and stop words, non-lemmatized	79.48%
With punctuation and stopwords, non-lemmatized	82.40%

Table 3.3: Advanced Text Handling

Configuration	Accuracy
Implementing OOV words handling and sentence contractions	83.86%

Table 3.4: Final Model with Hyperparameter Tuning and Architecture

Configuration	Accuracy
Hyperparameter tuning Bidirectional LSTM	85%

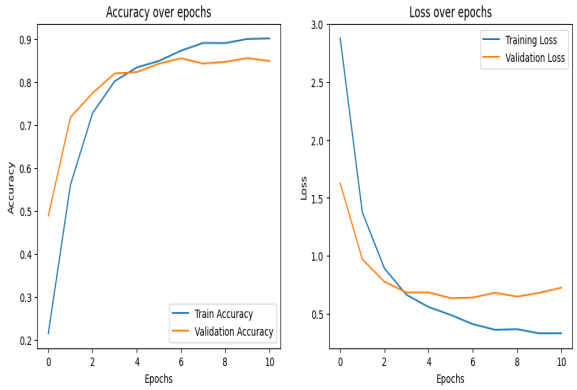


Figure 6: Final BiLSTM model - accuracy and loss over epochs

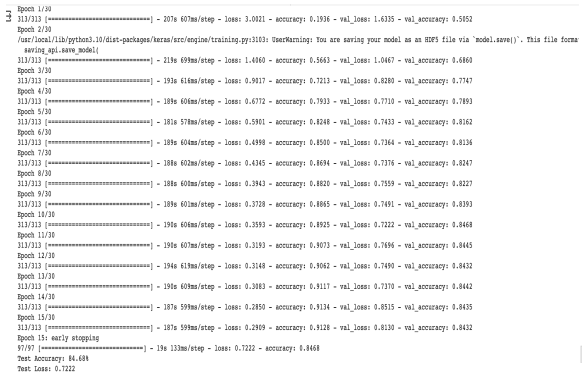


Figure 7: Training Over Epochs with final accuracy of 84.68%

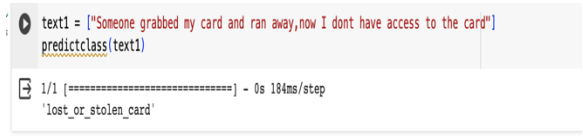


Figure 8: Predicting the intent with a text experiment

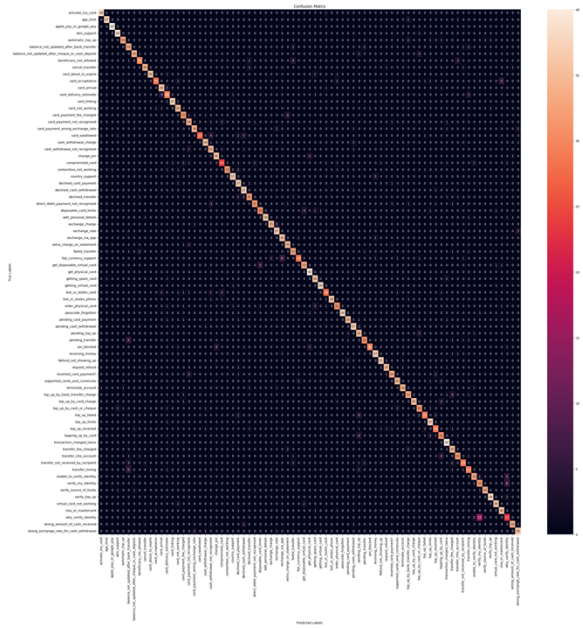
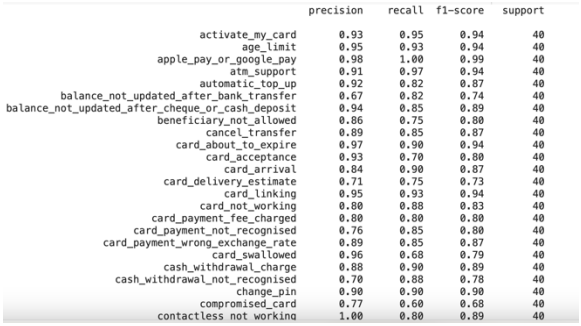


Figure 9: Heatmap of Confusion Matrix



	precision	recall	f1-score	support
activate_my_card	0.93	0.95	0.94	40
age_limit	0.95	0.93	0.94	40
apple_pay_or_google_pay	0.98	1.00	0.99	40
atm_support	0.91	0.97	0.94	40
automatic_top_up	0.92	0.82	0.87	40
balance_not_updated_after_bank_transfer	0.67	0.82	0.74	40
balance_not_updated_after_cheque_or_cash_deposit	0.94	0.85	0.89	40
beneficiary_not_allowed	0.86	0.75	0.80	40
cancel_transfer	0.89	0.85	0.87	40
card_about_to_expire	0.97	0.90	0.94	40
card_acceptance	0.93	0.70	0.80	40
card_arrival	0.84	0.90	0.87	40
card_delivery_estimate	0.71	0.75	0.73	40
card_linking	0.95	0.93	0.94	40
card_not_working	0.80	0.88	0.83	40
card_payment_fee_charged	0.80	0.80	0.80	40
card_payment_not_recognised	0.76	0.85	0.80	40
card_payment_wrong_exchange_rate	0.89	0.85	0.87	40
card_swallowed	0.96	0.68	0.79	40
cash_withdrawal_charge	0.88	0.90	0.89	40
cash_withdrawal_not_recognised	0.70	0.88	0.78	40
change_pin	0.90	0.90	0.90	40
compromised_card	0.77	0.60	0.68	40
contactless_not_working	1.00	0.80	0.89	40

Figure 10: Classification Report

4. Discussion of Results

4.1 Text Preprocessing (Table 3.1 and Table 3.2)

Keeping punctuation and common words like "the" and "is" in the sentences helped the model understand and categorize banking questions better. For example, when we didn't change the form of words to their simplest base form, the program got better at figuring out what people were asking. Usually, simplifying words helps a program learn general patterns, but in our tests, keeping the words as they were originally used was more helpful. This is likely because the exact way people phrase their questions is important for understanding the specific banking issues they're asking about.

4.2 Handling of Out-of-Vocabulary (OOV) Words (Table 3.3)

We got better results, up to 83.86% accuracy, when we found ways to deal with rare words and stretched out shortened words like “can’t” to “cannot.”

4.3. Model Architecture and Hyperparameter Optimization (Table 3.4)

Using a BiLSTM model helped because it looks at the information before and after a word to understand its meaning better. This made the model better at figuring out what people were asking about because it could see the full picture of each question. Adjusting the model's settings was important. It's like fine-tuning a car to perform well under different conditions. We changed these settings to help the model learn just right — not too little, not too much. This way, it got better at recognizing a wide variety of customer questions without getting confused.

4.4 Interpretation of Graph (Figure 6)

The Accuracy Over Epochs graph has two lines, one for training accuracy and one for validation accuracy. Over the epochs both lines go up, which means the model is getting better at making the right predictions as it practices more. They also stay close together, which is good because it suggests that the model is just as good with new examples as it is with the ones it practiced on. The Loss Over Epochs Graph have two lines showing how much error the model makes during training and validation. Both lines trend downwards, which is a sign that the model is making fewer mistakes over time. Just like with the accuracy graph, the fact that these lines are close together is good. It means the model isn’t just memorizing the training examples but learning to apply what it's learned to new situations.

4.5 Model Training Progress and Final Performance Metrics (Figure 7)

The training log shows that as the model went through more training epochs, it got better at predicting accurately, as seen by the increasing accuracy, and decreasing loss. The final accuracy of 84.68% and loss values suggest that the model learned effectively.

4.6 Classifying a Lost or Stolen Card Scenario (Figure 8)

A text input — "Someone grabbed my card and ran away, now I don't have access to the card" — is given to the model, which then predicts the label of this input as "lost_or_stolen_card". The model has processed the input and correctly identified it as a case of a lost or stolen card, which indicates that it has learned to associate certain phrases with specific intent categories effectively.

4.7 Intent Classification Confusion Matrix (Figure 9)

The diagonal line, which is notably bright, represents correct predictions where the model's predicted intent matches the true intent. The dark squares indicate very few misclassifications, which is good. However, there's a noticeable red square away from the diagonal, indicating a higher number of a particular type of error for a specific intent. This spot shows where the model is most often getting confused and incorrectly predicting a certain intent.

5. Conclusion

The results clearly indicate that preprocessing text by retaining punctuation and stop words improves the model's understanding of the context, thus improving accuracy. This could be because customer queries are generally short and removing stop words may thus make it difficult for the model to learn. The inclusion of lemmatization and handling of OOV words further refines the model's capability to classify intents more effectively the implementation of a bidirectional LSTM architecture with optimized hyperparameters in the final model configuration achieved the highest accuracy.

The findings show that paying close attention to how you prepare your text and set up your model can make a big difference in how well an LSTM works. By moving from simple to more complex setups, it’s clear that LSTMs have a lot to promise, especially if they are paired with good text preparation and neural network architectures. This can lead to very accurate systems for figuring out customer intentions in banking.

6. References

[1] BERT: Pre-training of Deep Bidirectional Transformers for Language.
https://arxiv.org/abs/1810.04805

[2] Improving Language Understanding by Generative Pre-Training
https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf

[3] Understanding Bidirectional LSTM for Sequential Data Processing
https://medium.com/@anishnama20/understanding-bidirectional-lstm-for-sequential-data-processing

[4] A Guide to Hyperparameter Tuning: Enhancing Machine Learning Models
https://medium.com/@abelkuriakose/a-guide-to-hyperparameter-tuning-enhancing-machine-learning-models-69dc9e0f02ea