Intent Classification for Bank Customer Queries

Yumi Heo
230003122
MSc Data Science
yumi.heo@city.ac.uk

1 Problem statement and Motivation

We are increasingly accustomed to interacting with chatbots instead of directly connecting with counsellors on company websites and mobile apps, regardless of whether the company sells tangible products (e.g., e-commerce) or intangible services (e.g., financial products). While chatbots may not provide nuanced responses, they can determine the category most closely related to a user's inquiry and then either deliver an answer or direct the user to a relevant link. This study aims to understand how these chatbots classify user inquiries into internally designated categories. We will use the Hugging Face banking77 dataset, which has 77 classes. In real-world scenarios, each company will define the number of classes based on their specific business needs. However, by examining this dataset, we can gain a fragmented glimpse into how language data is classified for chatbot applications.

Before the advent of Transformers, RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory) were popular model structures in natural language processing (NLP). However, the BERT (Bidirectional Encoder Representations from Transformers) model, which appeared in 2018, became a game changer in NLP research. This study will investigate the performance of LSTMs with three different word embedding types. We will then compare the performance and efficiency of these LSTMs to DistilBERT, a smaller, faster version of BERT. Assuming companies have infinite time and resources to spend, having a model that produces the best possible performance and has a positive impact on business would be the best option for companies. However, in reality, there is a high possibility of choosing a model that is cost and time-efficient and has the best performance, or a model that has persuasive performance even if it is not the best. In conclusion, through this study, we will understand how LSTM and DistilBERT perform multi-class classification, which model shows the best performance, and which model is efficient.

2 Research hypothesis

To achieve the goals outlined above, this study establishes two key hypotheses:

DistilBERT Performance: The DistilBERT model will show the best overall performance among the compared models. However, this advantage will come at the cost of a larger model size and longer training time.

LSTM Model Size: The LSTM model using Keras' built-in embedding layer will have the smallest size amongst all LSTM models. This is because the other LSTM models will obtain pre-trained word embeddings such as Word2Vec or GloVe, which require importing files from outside of the baseline model. Conversely, the Keras model configures its own embedding layer within the model itself.

By proving these two hypotheses through research, we will be able to determine which model is more appropriate from a business perspective.

3 Related work and background

The LSTM model was created to solve the vanishing and exploding gradients that occur in backpropagation, which RNN had difficulty with (Hochreiter & Schmidhuber, 1997). It was able to analyze relatively long range sequences than RNN and worked efficiently with the gradient algorithm, but until BERT appeared in 2018, LSTM also had the problem of vanishing gradient.

BERT, using a transformer structure, cannot have the vanishing gradient problem because this model does not follow typical neural networks. According to Devlin et al. (2018), this model, like the

Transformer model, processes the entire sequence at once using attention mechanisms Attention allows the model to focus on relevant parts of the sequence when processing each element, reducing the reliance on long-term dependencies through sequential processing (Devlin et al., 2018). However, since our study does not plan to classify relatively long text data and the dataset size is 392 KB, we use DistilBERT, a lightweight model of BERT. Moreover, because the resources given in this study are limited, using DistilBERT can reduce computational costs compared to BERT and train the model while taking up less memory capacity (Sanh et al., 2019).

However, there is a study that suggests that the size of the banking77 dataset used in our study is insufficient for model training. Loukas et al. (2023) conducted data augmentation to supplement the amount of data using GPT-4. There is another study that performed augmentation to lengthen short sentences in connection with intent analysis as we do for this study (Dopierre et al., 2021). However, regardless of the size of banking77, there is a research from PolyAI for model development for intent analysis. Casanueva et al. (2020) represented their new model which performs better than fine-tuned BERT and which is in a stable state even though hyperparameter settings are changed using banking77 dataset for intent analysis. However, although increasing the insufficient amount of data in intent analysis and creating a better model are meaningful and important studies, it is also necessary to understand what parts are missing in existing intent analysis. Zhang et al. (2021) suggested the importance of in-domain out-of-scope detection for intent analysis, and explained the vulnerability of pre-trained transformer models. There are numerous other studies available for reference on the datasets we use for this study or intent analysis, but we will focus on comparing LSTM when combined with various word embeddings in terms of performance and efficiency with DistilBERT. We plan to use Word2Vec and GloVe as pre-trained word embedding. Here, Word2Vec is a representative method of quantifying the meaning of words to reflect meaningful similarities between word vectors (Mikolov et al., 2013). The method of vectorizing the meaning of a word into a multidimensional space is called distributed representation, and is basically a representation method created under the assumption of the distributional hypothesis (Mikolov et al., 2013). GloVe is a word embedding methodology developed at Stanford University in the United States in 2014 that uses both count-based and prediction-based methodologies (Pennington et al., 2014). This was intended to point out and complement the shortcomings of prediction-based Word2Vec (Pennington et al., 2014). However, through this study, we will find out which is superior between Word2Vec and GloVe.

The development of these neural networks and transformer models had a significant impact on financial services. In particular, when considering financial services, one might typically think of time series-specific models, such as price prediction and exchange rate prediction. However, we will gain insight into the impact that language models have had from the perspective of businesses that provide financial services to the general public, such as banks and fintech companies. In particular, as LLM research for the finance sector, called FinLLMs, continues (Lee et al., 2024b), our research will focus on the differences in performance and efficiency between LSTM and DistilBERT on a dataset with financial characteristics called banking77.

4 Accomplishments

- Task 1: Data Preprocessing – Completed
- Task 2: Text Tokenising – Completed
- Task 3: Build and train baseline LSTM models – Completed
- Task 4: Build derivative models from the baseline for LSTM– Completed
- Task 5: Hyperparameter tuning of the baseline LSTM models – Completed
- Task 6: Hyperparameter tuning of the derivative LSTM models – Failed due to time limitations
- Task 7: Test and perform error analysis for all LSTM models built – Completed
- Task 8: Build and train baseline DistilBERT models – Completed
- Task 9: Build derivative models from the baseline for DistilBERT – Failed due to time limitations
- Task 10: Hyperparameter tuning of the baseline DistilBERT models – Failed due to time limitations

- Task 11: Test and perform error analysis for all LSTM models built – Half completed, error analysis could not be performed due to time limitations
- Task 12: Test the best models for LSTM and DistilBERT with inference – Failed due to time limitations

5 Approach and Methodology

First, we need to compare the features of LSTM and DistilBERT. LSTM was designed to overcome the vanishing gradient problem, which was a drawback of RNN (Hochreiter & Schmidhuber, 1997). This is useful for learning long-term dependencies by managing short-term and long-term memory. However, as the model structure becomes more complex, computational costs increase. Additionally, when dealing with relatively long sequences, LSTM can suffer from gradient vanishing, as RNN does.

DistilBERT is a pretrained model that is lighter than BERT (Sanh et al., 2019). Therefore, it can be said to be more efficient than BERT as it uses less memory (Sanh et al., 2019). Since it is a model derived from BERT, it is useful for various natural language processing tasks through transfer learning using a large text corpus. However, the limitations of the transformer model are still the long training time required if GPU is not used and the larger memory usage compared to LSTM, which can be said to be disadvantages.

Before comparing DistilBERT and LSTM models, we decided which word embedding to use and then attached it to each separate LSTM's embedding layer. Most of the word embeddings used this time were pre-trained, with the first one being from Keras' embedding layer, which is not pre-trained but implements words from the training data and learns them as embedding vectors. For the second, we used pre-trained Word2Vec. The corpus file for Word2Vec was obtained from a downloadable file (GoogleNews-vectors-negative300.bin.gz). We then created a Word2Vec model using the gensim library, one of the Python libraries, and had the model learn the downloaded corpus. Lastly, we used GloVe, which was also pre-trained, and we decided to use the smallest file of 822 MB with 400,000 words among the corpus files posted on Stanfordnlp (n.d.) for this study. Both Word2Vec and GloVe contrast the first method, which learns from scratch using the existing training data as an embedding layer. Three types of word embeddings were used in each LSTM to create separate models, and then the performance of each model was compared. LSTM was built around the tensorflow.keras library to utilize the 'Embedding()' tool, Keras' embedding layer implementation.

Due to time constraints, we opted to use the DistilBERT model architecture without modifications when training, evaluating and comparing to the other models. However, we investigated the impact of data preprocessing on performance. While basic preprocessing was performed, including lowercasing and removing double whitespaces, numbers and punctuations, we opted against stemming or lemmatisation. Because the texts in our dataset are typically short and simple sentences assuming the limitations of customer inquiries, we decided that additional preprocessing would not be necessary. The reason to examine the difference with or without data preprocessing is that DistilBERT already has a layer that can accept the input data itself. Also, it is a pre-trained model taken from Hugging Face. Therefore, it would likely show higher performance on text that has not been preprocessed at all. This is to see the result whether it would indeed show higher performance for such pure text data. DistilBERT was implemented using the Hugging Face package with PyTorch. However, unlike LSTM, hyperparameter tuning was not performed on the DistilBERT model architecture, and detailed adjustments to learning rates or the number of layers were not carried out due to time limitations.

Whether each model was well trained was checked by plotting the training and validation loss and accuracy obtained after training. The performance of each LSTM and DistilBERT model was compared in their league first using test accuracy, precision, recall and F1 score. Also, for LSTM models, error analysis was conducted after testing their performance. However, error analysis for DistilBERT models was not conducted due to the time limitations. The model that showed the highest performance in LSTM and the DistilBERT model selected in this process were compared using a total of four evaluation indicators previously used. The best models from LSTM and DistilBERT were compared not only in terms of evaluation indicators but also in terms of training time and the size of the saved model file.

6 Dataset

The dataset used in this study is called 'banking77', provided by PolyAI, and it can be directly downloaded from Hugging Face to a code notebook using the 'datasets' package. Since it follows a Hugging Face dataset structure, it is divided into train and test sets, and the substructure comprises a dictionary divided into text and label. The provided training set consists of 10,003 data points, and the test set contains 3,080 text data points. There are no null data in this dataset. Additionally, there are no duplicate text data before preprocessing. However, it should be noted that duplicate text may occur depending on which type of preprocessing or which step of preprocessing was performed.

Training set	
Text	Label
I am still waiting on my card?	11 (card arrival)
I am inquiring about a $1 charge on my statement.	34 (extra charge on statement)

Table 1. Examples of the training set

In Table 1, the text of the given training set is a string. However, what stands out are numbers, punctuation, and special characters such as $. For reference, the label description in parentheses in Table 1 is added here. The label itself is an integer, which is from 0 to 76. When checking the distribution of each label in the training set and test set, we see the training set is imbalanced. In particular, the data classified into class 6 is the largest, and the smallest is class 23, which is about 6 times the amount of data compared to class 6 (Figure 1).

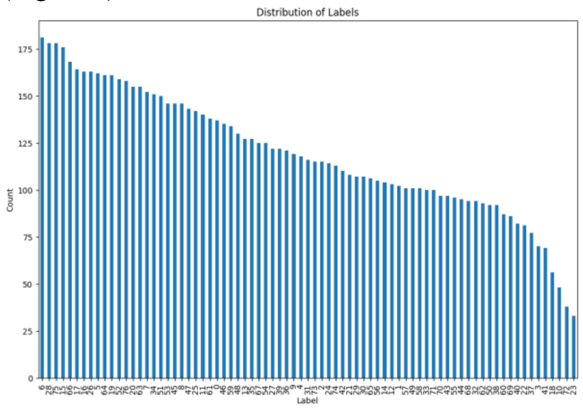

Figure 1. Imbalanced class in training set

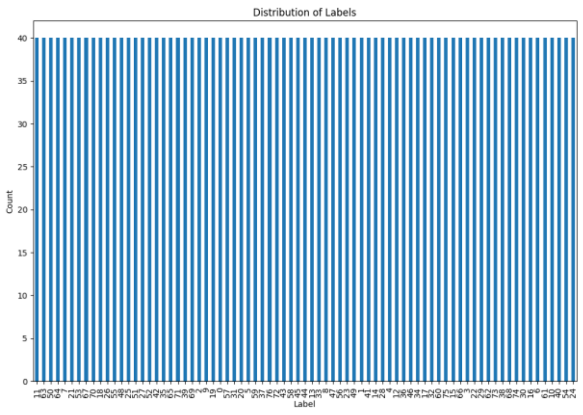

Figure 2. Balanced class in test set

However, when checking the class distribution of the test set, all classes have 40 data points equally (Figure 2).

The 'banking77' dataset has a dictionary-type hugging face-specific dataset structure, which makes it challenging to figure out how to proceed with preprocessing. There are two ways: directly apply custom preprocessing functions to the data set and preprocess it through mapping, or convert it to a dataframe and proceed with preprocessing. This part will be covered further in the following preprocessing part.

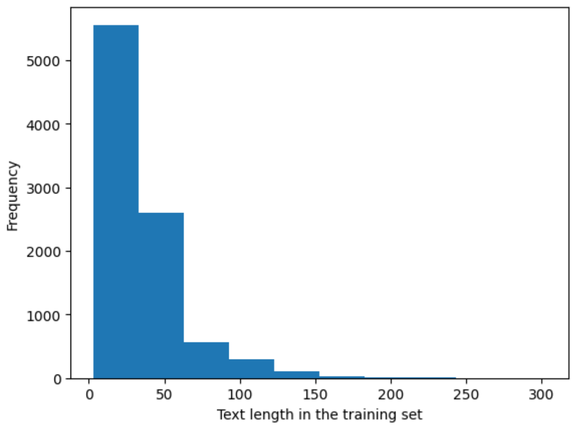

Figure 3. Text length in the training set

The text length of the training set was identified for padding after text tokenization. This was expressed as a histogram (Figure 3), and as a result of additional statistics, the average is around 36, the median is 29 and the maximum length is 303. Which numbers were used for padding will be discussed in the next part.

6.1 Dataset preprocessing

In all LSTM models, we removed numbers, punctuation, double white spaces, and stopwords, and applied lowercase. One of the DistilBERT models was also preprocessed to compare performance based on the presence or absence of data preprocessing. To conduct each preprocessing step, the 're' package was utilized to remove numbers and double white spaces, while the 'string' package was used to remove punctuation, including special characters. Additionally, the 'nltk' package was employed to delete stopwords. Finally, we converted all characters to lowercase using Python's built-in method, lower(). All of these processes were carried out in the Hugging Face dataset structure. Thus, each preprocessing function was applied to the entire dataset through mapping.

Next, from the dataset, which originally consisted of only the training set and test set, a validation set was created by separating it at a rate of 20% from the training set. This step was applied to both LSTM and DistilBERT models.

For tokenization, different methods were used depending on LSTM and DistilBERT. LSTM used the Tokenizer provided by tensorflow.keras, and DistilBERT utilized tokenization using DistilBertTokenizer in transformers provided by Hugging Face. When performing padding afterwards, the maximum length was set to 29. Here, 29 is the median value found in the training set before being separated into the validation set.

7 Baselines

The baseline model structure of LSTM is divided into three types according to word embedding: the Keras embedding layer, Word2Vec, and GloVe. All models have a total of three layers. The first layer of all three models is an embedding layer, and the only difference is the embedding dimension. The next layer is LSTM, with the hidden unit set to 30, a random number. The last layer is the dense layer, which must classify 77 classes as the final output layer. Here, the activation function is softmax for multi-class classification. All optimizers are Adam, and the learning rate is the default value of 0.001. Cross-entropy is used as the loss function. Since the label is a list consisting only of integers without one-hot encoding, Keras provides ‘sparse_categorical_crossentropy’ for this case. After setting up the initial model structure, the dropout is applied to the LSTM layer. Dropout is a type of regularization that randomly sets some hidden units to 0 when training a model (Team, K. (n.d.). Keras documentation: EarlyStopping. https://keras.io/api/callbacks/early_stopping/). A ratio of 0.2 is applied to randomly set the unit to 0. Next, hyperparameter tuning is performed to find the optimal value of the number of hidden units and the learning rate. The model is trained once again using the optimal value found through this hyperparameter tuning process, and dropout is also applied and the model is tested once more. This process is performed equally for a total of three LSTM models depending on the type of word embedding. During all training, the epoch is set to 100, but to prevent overfitting, early stopping is applied to prevent running the entire epochs.

The model architecture of DistilBERT is constructed after loading the pre-trained model from Hugging Face. First, the loaded model is saved in the model architecture, and then the linear layer is made to have 768 dimensions of DistilBERT. Also, since 77 classes need to be classified, the output is set to 77. For forward pass, input IDs and attention mask are accepted from the tokenized vector and this input is passed through the model to obtain the result. Here, the last hidden state is extracted to obtain the result of the model, which can be called embedding. Here, we derive the unnormalized value, logit, and output it as a result. The epoch for DistilBERT is set to 5. In DistilBERT, unlike the LSTM experiment, hyperparameter tuning is not performed, additional layers are not added or detailed adjustments to existing layers are not performed. Although it has the same model structure, this model is trained using the training set with a different way of preprocessing.

These LSTM and DistilBERT models are ultimately used as language models to classify 77 classes.

8 Results, error analysis

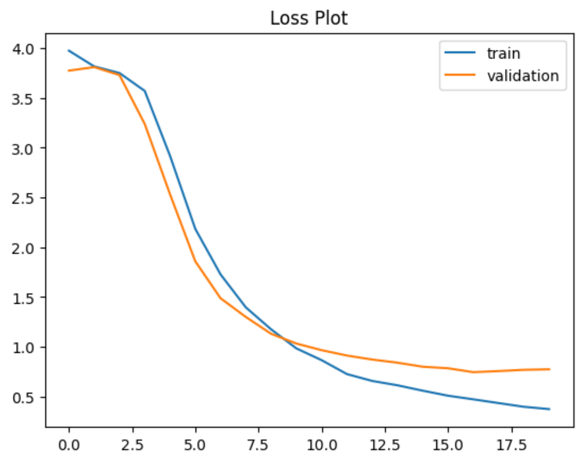

Figure 4. Best LSTM training loss

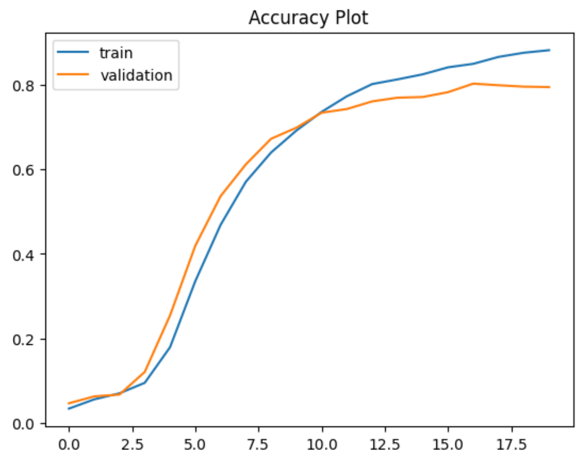

Figure 5. Best LSTM training accuracy

The test results of the three LSTM baselines, according to word embedding type, showed that the accuracy ranged between 50% and 68%, which did not demonstrate high performance. Through this, we found that changes were needed in the model structure, and as mentioned above, we searched for an optimal model through dropout and hyperparameter tuning. As a result, among the LSTM models, the model that used Word2Vec as embedding and applied dropout after hyperparameter tuning showed the highest performance. As can be seen in Figures 4 and 5, the model stopped after training only 20 epochs out of a total of 100 epochs due to the application of early stopping.

The proportion of data in the test set that this model misclassified was 20.16%, and it misclassified 621 pieces of data.

No.	Input text	Actual label	Predicted labed
1	locate card	11 (card_arrival)	13 (card_linking)
2	way know card arrive	11 (card_arrival)	12 (card_delivery_estimate)
3	get card	11 (card_arrival)	12 (card_delivery_estimate)
4	received card	11 (card_arrival)	12 (card_delivery_estimate)

Table 3. Error analysis for best LSTM

In Table 3, the model was noticeably incorrect in classifying the arrival, delivery time, location of the card or its specific use ('linking'). First, label 13 was confused with 11 due to the word 'locate'. And many of the labels 12 were confused with 11. Label 11 is a class about card arrival, but label 12 specifically inquires about the expected time of card arrival, so it may have been difficult for the model to classify. The model may have had difficulty in determining the label due to the short text length as some words and numbers were removed from preprocessing.

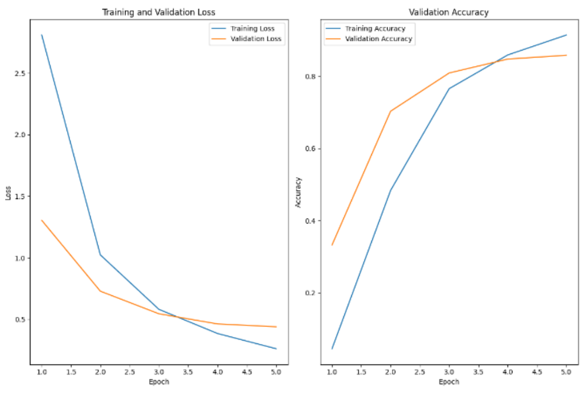

Figure 6. Best DistilBERT training loss and accuracy

DistilBERT went through the same training process without differences in model structure or learning rate. The only difference was whether preprocessed data was used or not, and the model that was ultimately confirmed to perform better was the model using preprocessed data. In Figure 6, the model was trained for a total of 5 epochs, and training was stopped before overfitting occurred.

	Accuracy	Precision	Recall	F1-score
Best LSTM model	79.84	81.46	79.84	79.54
Best DistilBERT model	89.55	90.08	89.55	89.56

Table 4. Evaluation indicators of best models

The comparison between LSTM's best model and the best DistilBERT was done through four indicators: accuracy, precision, recall and F1-score. The Best DistilBERT model shows results that are approximately 10 higher in all indicators (Table 4). Through this, we can confirm that DistilBERT, a transformer model, shows higher performance than that of the best LSTM model.

	Training time (sec)	File size (MB)
Best LSTM model	210.85 (with CPU)	3.3
Best DistilBERT model	24.42 (with T4 GPU)	506.8

Table 5. Additional indicators of best models

However, DistilBERT clearly increased model training speed exponentially when using CPU compared to the LSTM model. The training progressed at a level that could not be measured in seconds, so in the end, T4 GPU was used, provided free of charge by Google Colab. For a fair comparison, the training time could have been measured when the LSTM model was training using GPU, but due to GPU resource limitations in free accounts, GPU was used only for DistilBERT training. If this challenge is viewed from a business perspective, the transformer model may be difficult to adopt when looking for a model that shows maximum performance at minimum cost. However, if model training is carried out with separate external resources and the saved model continues to be used using internal resources, the standards for resource consumption may vary. Nevertheless, when comparing the size of the saved model, DistilBERT was 169 times larger than LSTM (Table 5). Therefore, the company will have to decide whether to use the best LSTM model found in this study for relatively low-cost and efficient storage management, developing it a little further, or to use the DistilBERT model by investing a little more in training, storage, and usage costs.

9 Lessons learned and conclusions

Through this study, it was proven that the transformer-based model shows the highest level of test performance. Even though the classes in the training set are imbalanced, DistilBERT demonstrated accuracy and F1 scores close to 90 after relearning with a refined training set. However, the LSTM model might show the best performance after further development. If the imbalance in the training set is adjusted through data augmentation, or cross-validation is applied, since the dataset size is small, there is potential for both models to demonstrate better performance than they currently do. Additionally, if fine-tuning is conducted for each layer in the structure of both models, this could also lead to performance improvement. For example, LSTM could be tested later by applying bidirectional LSTM, and DistilBERT could have an additional layer added. Moreover, experimenting with various learning rates for each model may lead to obtaining a better model than the current state.

We found that the transformer model has high storage costs as it shows high performance. Therefore, continuous research will be needed to determine how to lighten deep learning models through fine-tuning. For example, we could consider adapting knowledge distillation, which means the knowledge of a large trained model is sent to another smaller model to achieve the effect of model compression without losing its performance (Hinton et al., 2015).

	LSTM (keras embedding layer)	LSTM (Word2Vec)	LSTM (GloVe)
File size (MB)	2.7	3.3	1.2

Table 6. File size of LSTM models

We experimented with LSTMs using various word embedding methods and found that the model using LSTM with GloVe had the smallest size. In particular, when comparing models that applied dropout after hyperparameter tuning, in Table 6, the file size of LSTM with GloVe is about twice as small as that of LSTM with Word2Vec. The assumption is that even though the GloVe file itself was large before training, not all words were matched when compared with the vocabulary dictionary extracted from the training set. This suggests that the word embedding size was relatively small compared to that in other models. For example, the word 'topup' in the vocabulary dictionary did not match in GloVe. However, this part requires additional study to find out why the size of the LSTM model using GloVe, pre-trained through an external file, has the smallest size compared to the other two models.

Language models require inference after testing to confirm whether the model is running properly, but the inference part was excluded in this study. Therefore, if the inference of the currently available best model is carried out through the API provided by Hugging Face after the additional studies, or if the opportunity is given, more bidirectional interaction between users and models can be expected.

References

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1810.04805

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1910.01108

Loukas, L., Stogiannidis, I., Diamantopoulos, O., Malakasiotis, P., & Vassos, S. (2023). Making LLMs Worth Every Penny: Resource-Limited Text Classification in Banking. 4th ACM International Conference on AI in Finance, 392–400. https://doi.org/10.1145/3604237.3626891

Dopierre, T., Gravier, C., & Logerais, W. (2021). ProtAugment: Unsupervised diverse short-texts paraphrasing for intent detection meta-learning. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2105.12995

Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., & Vulić, I. (2020). Efficient Intent Detection with Dual Sentence Encoders. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2003.04807

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. M. (2013). Efficient estimation of word representations in vector space. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1301.3781

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 1532–1543. https://doi.org/10.3115/v1/D14-1162

Zhang, J., Hashimoto, K., Wan, Y., Liu, Z., Liu, Y., Xiong, C., & Yu, P. S. (2021). Are pretrained transformers robust in intent classification? a missing ingredient in evaluation of Out-of-Scope intent detection. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2106.04564

Lee, J., Stevens, N., Han, S. C., & Song, M. (2024). A survey of Large Language Models in Finance (FINLLMs). arXiv (Cornell University). https://doi.org/10.48550/arxiv.2402.02315

Stanfordnlp. (n.d.). GitHub - stanfordnlp/GloVe: Software in C and data files for the popular GloVe model for distributed word representations, a.k.a. word vectors or embeddings. GitHub. https://github.com/stanfordnlp/GloVe?tab=readme-ov-file

Team, K. (n.d.). Keras documentation: EarlyStopping. https://keras.io/api/callbacks/early_stopping/

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. https://doi.org/10.48550/ARXIV.1503.02531