UNIVERSITÄT BERN

CAS in Advanced Machine Learning

University of Bern

Question Classification Using Transformer Models and LLM-based QA system.

submitted to

Faculty of Science

Institute of Mathematics

submitted by

Gözde Özdemir

goezde.oezdemir@students.unibe.ch

Matriculation Number: 23-133-028

&

Sonam Lhamu

sonam.lhamu@students.unibe.ch

Matriculation Number: 13-503-859

Bern, 14th of June 2024

Table of Contents

1 ABSTRACT **3**

2 INTRODUCTION **3**

3 EXPLORATORY DATA ANALYSIS **4**

3.1 DATASET DESCRIPTION 4

3.2 DATA VISUALIZATION AND STATISTICS 6

3.3 DATA PREPROCESSING AND CLEANING 9

3.4 DATA PREPARATION AND SPLITTING 10

3.4.1 Data preparation 10

3.4.2 Data splitting 12

3.4.3 Preprocessing for NLP models 12

4 MACHINE LEARNING ANALYSIS **12**

4.1 QUESTION CLASSIFICATION: MODEL ARCHITECTURE 12

4.2 Q&A SYSTEM: ARCHITECTURE AND IMPLEMENTATION 14

5 RESULT DISCUSSIONS **17**

6 CONCLUSION AND OUTLOOK **20**

7 ACKNOWLEDGMENTS **21**

1 Abstract

This study, conducted as part of the Certificate of Advanced Studies (CAS) in Advanced Machine Learning program at the University of Bern, focuses on enhancing the customer service experience at GS Bank, a hypothetical financial institution, through advanced question classification techniques. In the banking and finance sector, accurately categorizing customer inquiries is essential for improving response times, increasing customer satisfaction, and streamlining support processes.

Our research leverages pre-trained transformer models to classify client questions into specific categories such as Accounts, Cards, Insurance, and Loans. By automating this classification process, we aim to provide valuable insights into customer needs and optimize the bank's product offerings accordingly. Additionally, this study explores the integration of a retrieval-based Question Answering (QA) system using a Large Language Model (LLM) to further enhance the bank's responsiveness to client queries. Our results demonstrate the potential of combining machine learning models with practical banking applications to significantly improve customer support and product optimization strategies.

2 Introduction

The primary objective of this research is to develop a comprehensive strategy for GS Bank to enhance customer service by using advanced natural language processing techniques. This involves three key components:

- Question Classification: Systematically classify client questions into predefined categories using state-of-the-art NLP models like DistilBERT. By accurately categorizing customer queries, GS bank can improve customer service efficiency, automate support workflows, and gain valuable insights into customer needs and preferences.
- Optimization of Banking Products and Services: Analyzing categorized questions to gain insights into customer demands and preferences. This will enable the optimization of banking products and services, ensuring they are tailored to meet customer expectations and improve overall satisfaction.
- Implementation of a Large Language Model-based QA System: Integrating a retrieval-based Question Answering (QA) system utilizing a Large Language Model (LLM). This

system will provide accurate and timely responses to customer inquiries, further enhancing the customer service experience.

Developed in Python and Google Colab, our project focuses on leveraging transfer learning for question classifications by fine-tuning three pre-trained variants of BERT: DistilBERT, RoBERTa and ALBERT.
These models are advanced variants of the original BERT(Bidirectional Encoder Representation from Transformers) model, which was introduced by Google in 2018. BERT revolutionized natural language processing by using a transformer architecture to read text bidirectionally and employing two training objectives: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM randomly mask tokens in the input and tries to predict them, while NSP predicts whether two sentences are sequentially connected.

DistilBERT introduced by Hugging face in 2019 is smaller, faster version of BERT.
It uses a method called distillation where a smaller model learns to mimic a larger model.

RoBERTa (Robust Optimized BERT Pretraining Approach) introduced by Facebook in 2019 is an optimized version of BERT. It uses larger mini-batches and longer sequences during training. It was trained with more data and computational resources than BERT.

ALBERT(A Lite BERT) introduced by Google Research and Toyota Technical Institute of Chicago in 2019 is a lighter and memory efficient version of BERT. It shares parameters across layers and does sentence order prediction (SOP) instead of NSP to better model inter-sentence coherence.

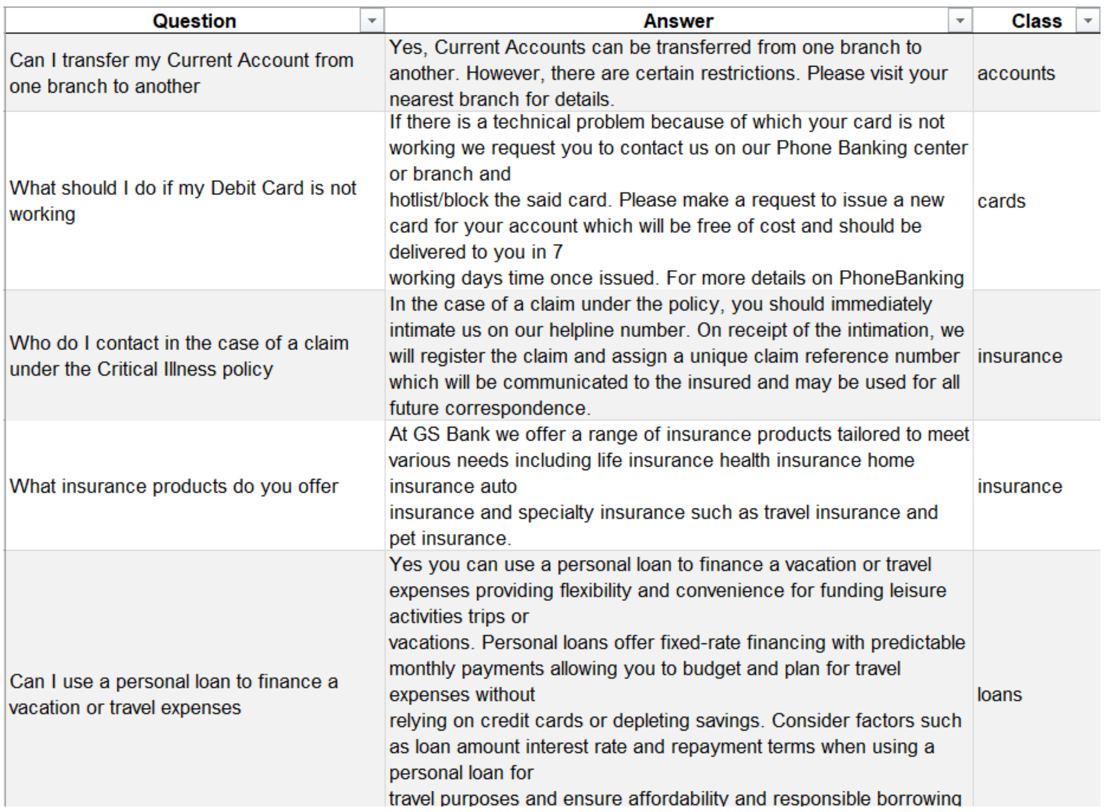
All necessary files and code for this project are accessible via our GitHub repository :
GitHub - commitit/FinalProject

3 Exploratory data analysis

3.1 Dataset description

Our input dataset is a CSV file consisting of three columns: “Question”, “Answer” and “Class”.

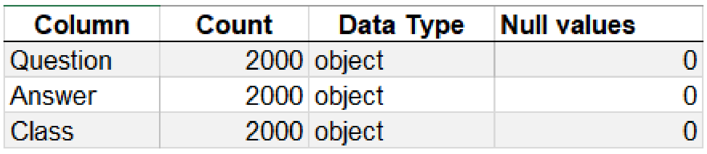
- Question: Questions customers might have about the different services provided by the bank.
- Answer: Corresponding answers to the questions.
- Class: Categories in which the corresponding questions-answer pair falls into. It is categorized into four classes: Loans, Insurance, Cards, Accounts.



Question	Answer	Class
Can I transfer my Current Account from one branch to another	Yes, Current Accounts can be transferred from one branch to another. However, there are certain restrictions. Please visit your nearest branch for details.	accounts
What should I do if my Debit Card is not working	If there is a technical problem because of which your card is not working we request you to contact us on our Phone Banking center or branch and hotlist/block the said card. Please make a request to issue a new card for your account which will be free of cost and should be delivered to you in 7 working days time once issued. For more details on PhoneBanking	cards
Who do I contact in the case of a claim under the Critical Illness policy	In the case of a claim under the policy, you should immediately intimate us on our helpline number. On receipt of the intimation, we will register the claim and assign a unique claim reference number which will be communicated to the insured and may be used for all future correspondence.	insurance
What insurance products do you offer	At GS Bank we offer a range of insurance products tailored to meet various needs including life insurance health insurance home insurance auto insurance and specialty insurance such as travel insurance and pet insurance.	insurance
Can I use a personal loan to finance a vacation or travel expenses	Yes you can use a personal loan to finance a vacation or travel expenses providing flexibility and convenience for funding leisure activities trips or vacations. Personal loans offer fixed-rate financing with predictable monthly payments allowing you to budget and plan for travel expenses without relying on credit cards or depleting savings. Consider factors such as loan amount interest rate and repayment terms when using a personal loan for travel purposes and ensure affordability and responsible borrowing	loans

Table 1: Sample data

Our dataset has 2000 entries, with no null values, equally distributed among the four classes, each with 500 entries.



Column	Count	Data Type	Null values
Question	2000	object	0
Answer	2000	object	0
Class	2000	object	0

Table 2: Data columns and data type

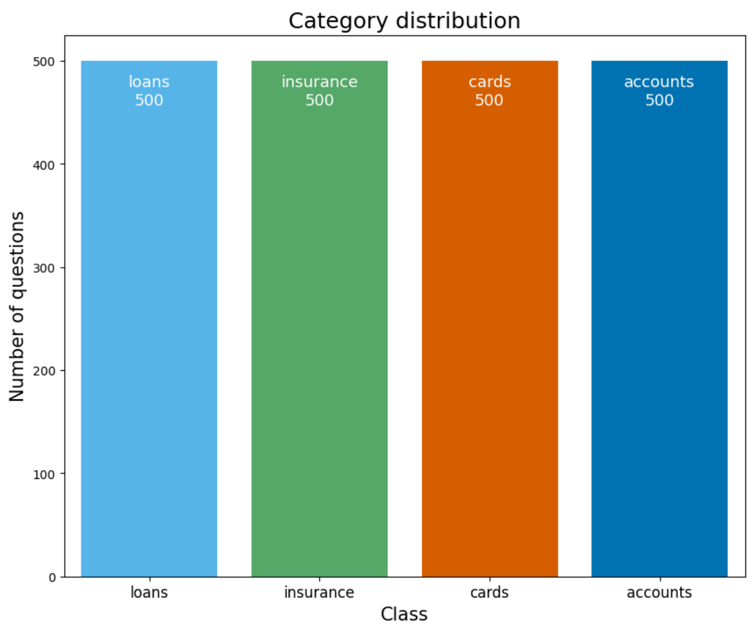
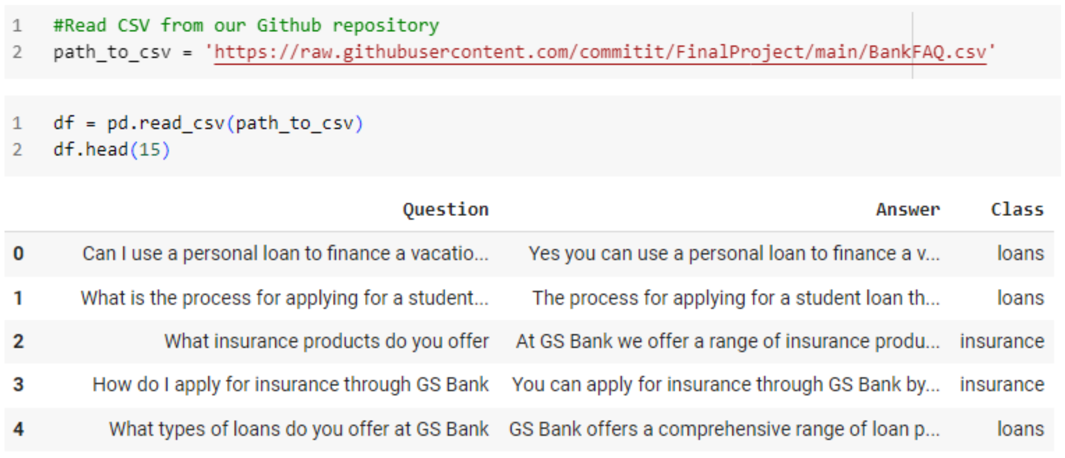


Figure 1: Data distribution.

3.2 Data visualization and statistics

Our dataset is well balanced between the different classes, now the next step is to check how balanced are the length of questions in our dataset. Many NLP models require a fixed-length input sequences, so knowing the length of the input sentences can help in padding or truncating sentences to a specific length. Some transformer models can in fact handle varying input lengths but still knowing the max length of our input sentences can help in optimizing our training and is fact an important parameter that impacts the model's performance.

Using pandas and matplotlib libraries, we visualized the distribution of word counts with the histogram and a violin plot below.

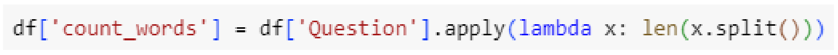


```
#Read CSV from our Github repository
path_to_csv = 'https://raw.githubusercontent.com/commitit/FinalProject/main/BankFAQ.csv'
```

```
df = pd.read_csv(path_to_csv)
df.head(15)
```

	Question	Answer	Class
0	Can I use a personal loan to finance a vacatio...	Yes you can use a personal loan to finance a v...	loans
1	What is the process for applying for a student...	The process for applying for a student loan th...	loans
2	What insurance products do you offer	At GS Bank we offer a range of insurance produ...	insurance
3	How do I apply for insurance through GS Bank	You can apply for insurance through GS Bank by...	insurance
4	What types of loans do you offer at GS Bank	GS Bank offers a comprehensive range of loan p...	loans

Figure 2: Load data



```
df['count_words'] = df['Question'].apply(lambda x: len(x.split()))
```

Figure 3: Calculate number of words

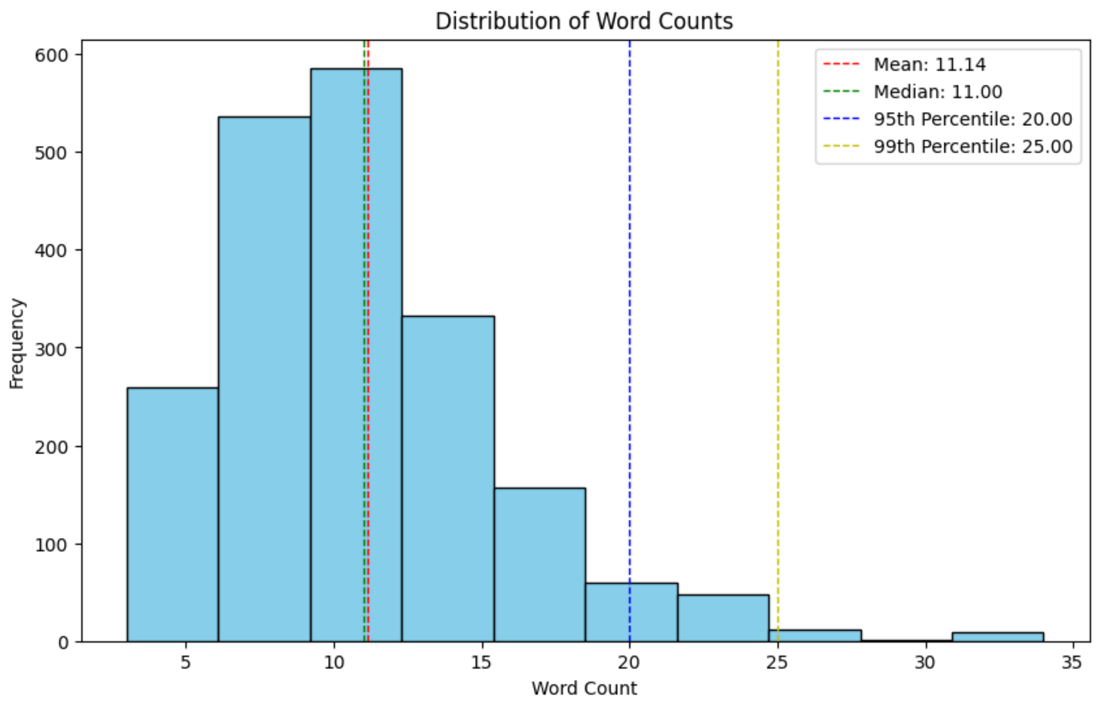


Figure 4: Histogram showing the number of words in each question.

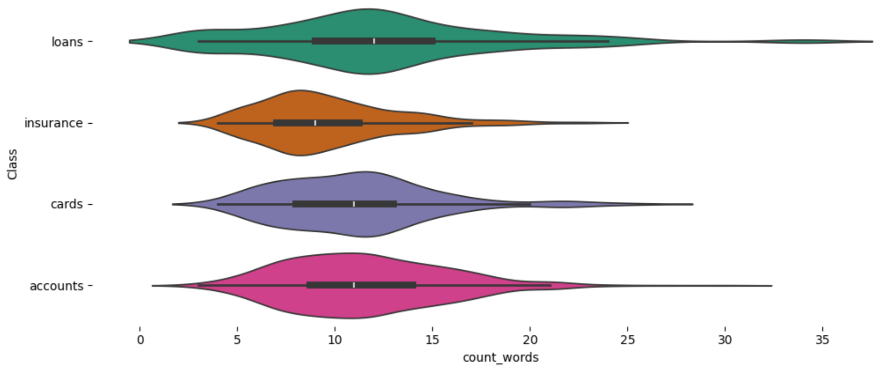


Figure 5: Violin plot to show the distribution of words within a class.

As seen in the visualization, most questions in our dataset are under 35 words and 200 characters, before data preprocessing, ensuring that our data will be well within the 512 token limit for our NLP models.

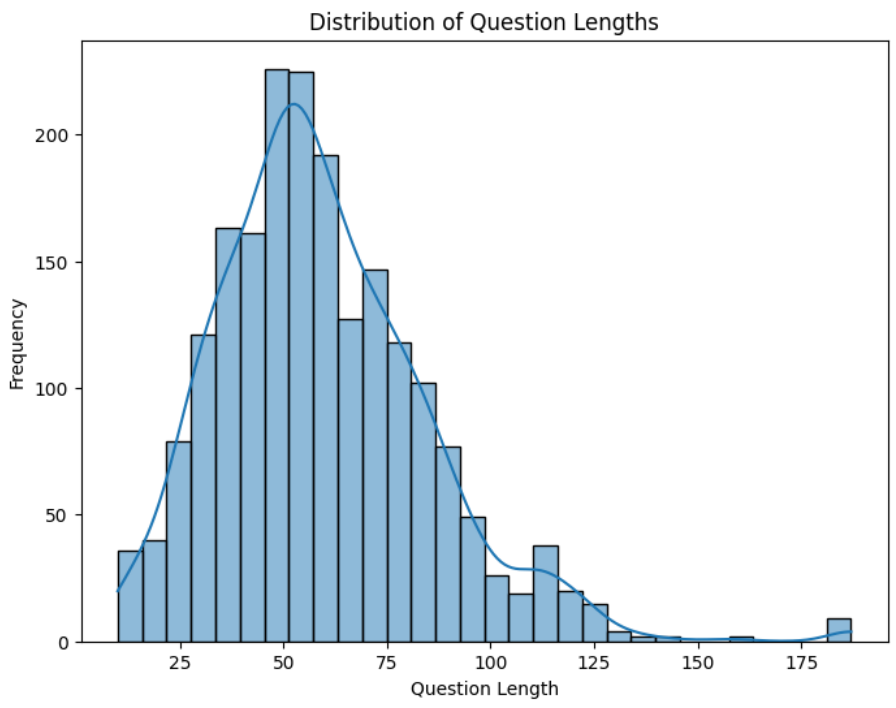


Figure 6: Histogram to show the distribution of characters in each question.

The visual representation below also shows that a lot of stop words are present in our dataset, which will be preprocessed and cleaned from our dataset before starting with the NLP model and training.

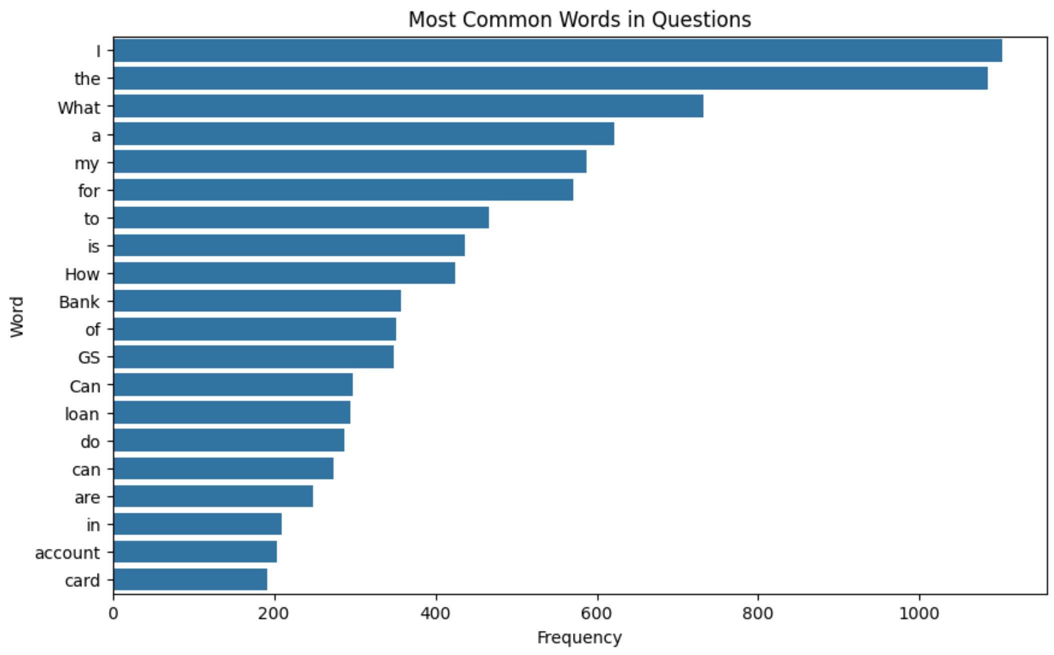


Figure 7: Chart showing the most common words in our dataset.

GS
Bank
loan
account
transaction
change
Debit Card
credit card
Current Account
policy
card
will
need
plan
option
obtain
apply
benefit
purchase
available
repayment account
Insurance policy
Fixed Deposit
ForexPlus Card
Education Loan
Savings Account
document

Figure 8: Word Cloud showing the most common words.

3.3 Data preprocessing and cleaning

Text preprocessing is a crucial step in preparing the data before we feed it to the NLP model. Our objective was to clean and normalize our question dataset and make it simpler for further data embedding and modeling.

Steps we performed:

1. **Remove HTML Tags**: HTML tags do not carry any meaningful information for our case, so by removing it we only keep relevant information.
2. **Conversion to lowercase:** we normalized all text to lowercase to avoid treating 'Bank' and 'bank' differently and thereby making the model perform better when the text data is consistent.
3. **Remove URLs:** URLs being present in our dataset, are not useful in building our model and to avoid further noise in the data we removed it from our text data.
4. **Remove all punctuations:** Even though punctuations could contain some contextual meaning, for our model it was just noise and not useful. So, we removed all punctuations from our "questions" data.
5. **Remove Stop words:** Lastly, with the help of the NLTK library, we removed all English stop words such as 'the', 'a', 'an' etc. Stop words are words that add little to the understanding of a text and ignoring such words will draw importance to the more important words in a text, while reducing the size of our data.

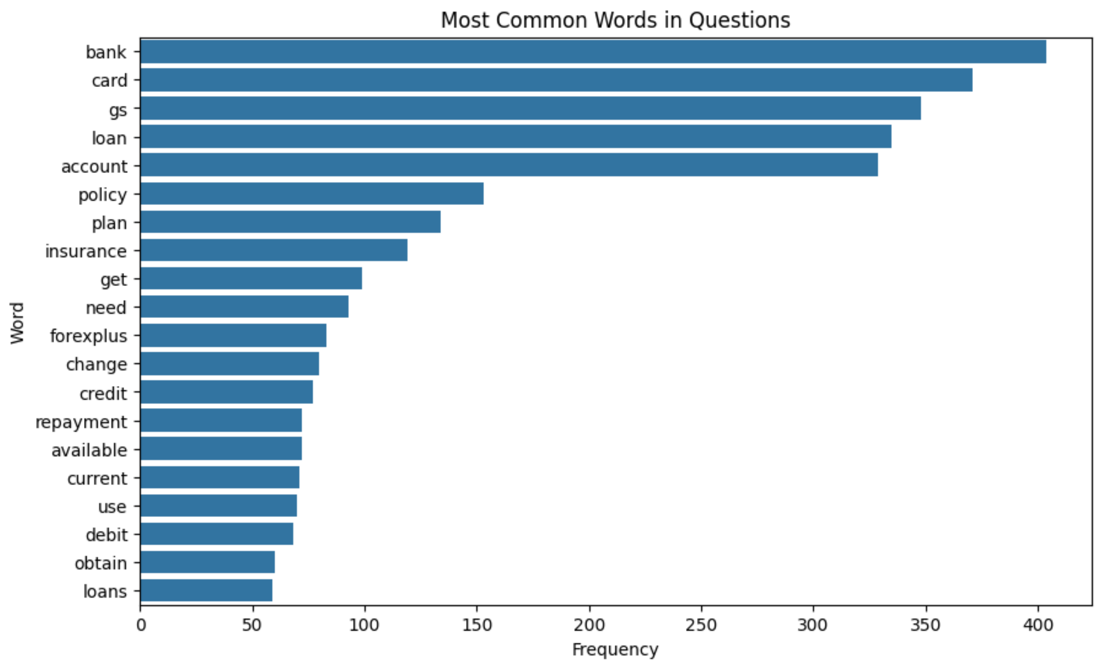


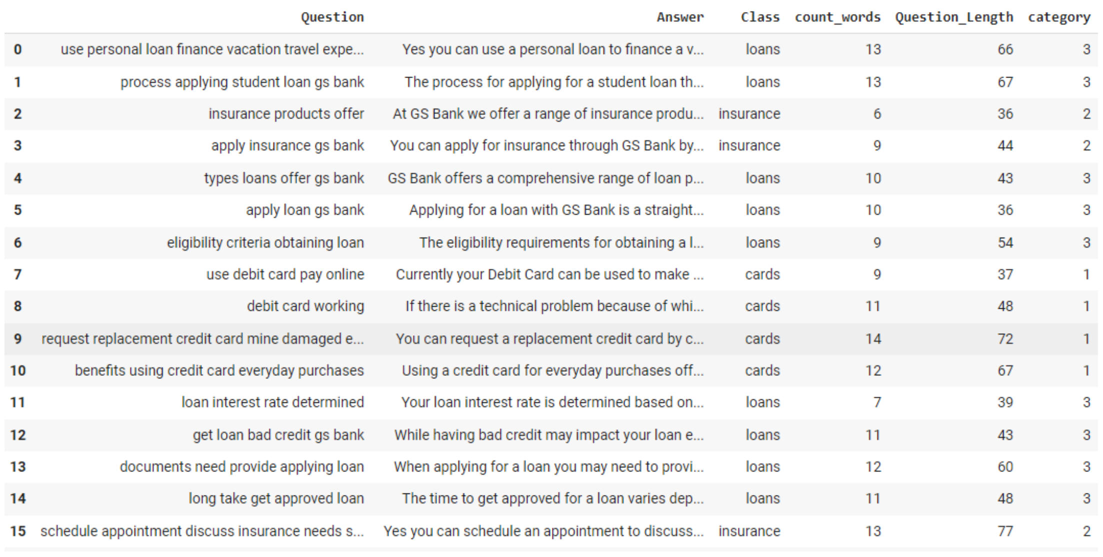
Figure 9: Chart showing most common words after data preprocessing.

3.4 Data preparation and splitting

3.4.1 Data preparation

1. **Label encoding**: For the first part of our project, our model will learn to predict the category of the question the client is asking. And since our categories are of object type,

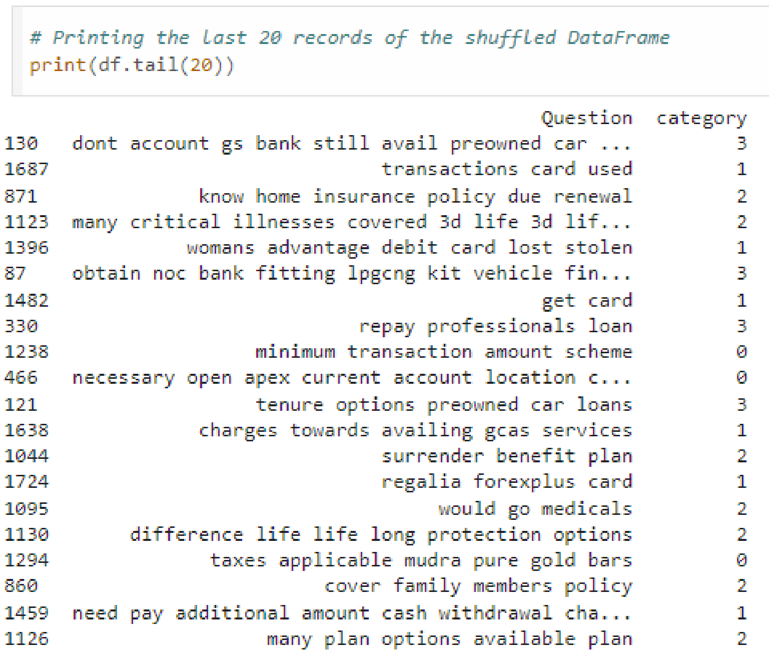
we need to convert it to a numerical representation before feeding it to our model to learn. We use label encoding from sklearn, assigning each class a unique integer from 0 to 3.



	Question	Answer	Class	count_words	Question_Length	category
0	use personal loan finance vacation travel expe...	Yes you can use a personal loan to finance a v...	loans	13	66	3
1	process applying student loan gs bank	The process for applying for a student loan th...	loans	13	67	3
2	insurance products offer	At GS Bank we offer a range of insurance produ...	insurance	6	36	2
3	apply insurance gs bank	You can apply for insurance through GS Bank by...	insurance	9	44	2
4	types loans offer gs bank	GS Bank offers a comprehensive range of loan p...	loans	10	43	3
5	apply loan gs bank	Applying for a loan with GS Bank is a straight...	loans	10	36	3
6	eligibility criteria obtaining loan	The eligibility requirements for obtaining a l...	loans	9	54	3
7	use debit card pay online	Currently your Debit Card can be used to make ...	cards	9	37	1
8	debit card working	If there is a technical problem because of whi...	cards	11	48	1
9	request replacement credit card mine damaged e...	You can request a replacement credit card by c...	cards	14	72	1
10	benefits using credit card everyday purchases	Using a credit card for everyday purchases off...	cards	12	67	1
11	loan interest rate determined	Your loan interest rate is determined based on...	loans	7	39	3
12	get loan bad credit gs bank	While having bad credit may impact your loan e...	loans	11	43	3
13	documents need provide applying loan	When applying for a loan you may need to provi...	loans	12	60	3
14	long take get approved loan	The time to get approved for a loan varies dep...	loans	11	48	3
15	schedule appointment discuss insurance needs s...	Yes you can schedule an appointment to discuss...	insurance	13	77	2

Table 3: Sample data after text preprocessing and label encoding.

2. **Remove unused columns:** We retain only "Question" and "Category" and drop the others from our current data set.
3. **Shuffle data:** Before we move on to splitting the dataset into training and test data, we also need to shuffle the data. For us it was important to ensure that the data is randomly mixed to prevent any order-related biases during training and testing.



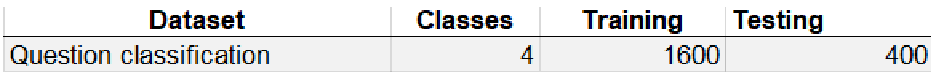
```python
# Printing the last 20 records of the shuffled DataFrame
print(df.tail(20))
```

```
                                               Question  category
130   dont account gs bank still avail preowned car ...         3
1687                              transactions card used         1
871                  know home insurance policy due renewal         2
1123  many critical illnesses covered 3d life 3d lif...         2
1396                    womans advantage debit card lost stolen         1
87    obtain noc bank fitting lpgcng kit vehicle fin...         3
1482                                            get card         1
330                                   repay professionals loan         3
1238                        minimum transaction amount scheme         0
466   necessary open apex current account location c...         0
121                            tenure options preowned car loans         3
1638                    charges towards availing gcas services         1
1044                                  surrender benefit plan         2
1724                                  regalia forexplus card         1
1095                                       would go medicals         2
1130          difference life life long protection options         2
1294                     taxes applicable mudra pure gold bars         0
860                               cover family members policy         2
1459  need pay additional amount cash withdrawal cha...         1
1126                          many plan options available plan         2
```

Figure 10: Shuffled sample data with only the necessary columns.

3.4.2 Data splitting

We decided to split the data into 80% training and 20% test data.



Dataset	Classes	Training	Testing
Question classification	4	1600	400

Table 4: Train and Test data split

3.4.3 Preprocessing for NLP models

Tokenization:

Before we fine-tune each of our models, it is essential to tokenize each of our input questions using the respective tokenizer for each model.

Tokenization is a crucial preprocessing step for NLP tasks. It is a process of converting texts into smaller units called tokens and assigning a numerical value to each token. Most NLP models require input in the form of tokens, as each model has their own predefined vocabularies and can only process text that has been tokenized as per their tokenization scheme.

We used the 'DistilBertTokenizerFast', 'RobertaTokenizer' and 'AlbertTokenizer' class for our respective model.

4 Machine Learning analysis

4.1 Question Classification: Model Architecture

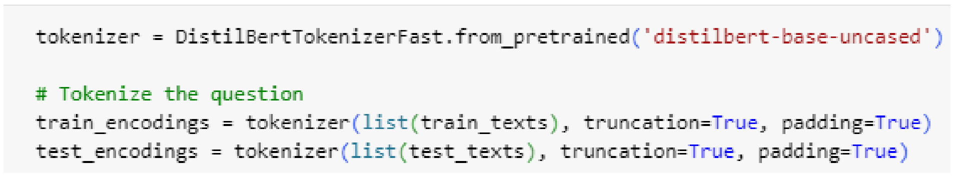
For this study, we utilized the DistilBERT, RoBERTa and ALBERT model for the task of question classification. The implementation involved several key steps to ensure effective training and performance optimization.

DistilBERT:

DistilBERT employs a technique called distillation, where a compact model is trained to replicate the behavior of a larger model. This distillation process allows DistilBERT to maintain approximately 97% of BERT's language understanding capabilities while being 60% faster and requiring only half the parameters.

1. Model Loading: We began by loading the pre-trained DistilBERT model designed specifically for sequence classification tasks. The DistilBertForSequenceClassification class from the Hugging Face Transformers library was used, which is tailored for classification problems.
2. Data Tokenization: Using the DistilBertTokenizerFast class, we tokenized our dataset, ensuring that the input text is converted into a format suitable for the model. The

tokenization process included padding and truncation to manage varying lengths of input sequences.



```
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')

# Tokenize the question
train_encodings = tokenizer(list(train_texts), truncation=True, padding=True)
test_encodings = tokenizer(list(test_texts), truncation=True, padding=True)
```

Figure 11: Code tokenizing our train and test data for DistilBERT.

RoBERTa:

RoBERTa enhances the training methodology of BERT by utilizing larger mini batches, longer training periods, and removing the next sentence prediction objective. These improvements allow RoBERTa to achieve superior performance on various NLP tasks.

1. Model Loading: Like DistilBERT, we loaded the pre-trained RoBERTa model tailored for sequence classification tasks using the RobertaForSequenceClassification class from the Hugging Face Transformers library.
2. Data Tokenization: The RobertaTokenizer was used to tokenize our dataset, ensuring the text is formatted correctly for the RoBERTa model. Padding and truncation were applied as needed.

ALBERT:

ALBERT achieves parameter reduction by sharing parameters across layers and employs a sentence order prediction (SOP) objective instead of the next sentence prediction (NSP) objective used by BERT. These modifications result in a more memory-efficient model with improved performance on downstream tasks.

1. Model Loading: The pre-trained ALBERT model designed for sequence classification was loaded using the AlbertForSequenceClassification class from the Hugging Face Transformers library.
2. Data Tokenization: The AlbertTokenizer was used to tokenize the dataset, ensuring compatibility with the ALBERT model. Padding and truncation were applied appropriately.

For all three models, we converted our tokenized data into PyTorch tensors and used the DataLoader class to facilitate batching, shuffling, and loading of the dataset during training.

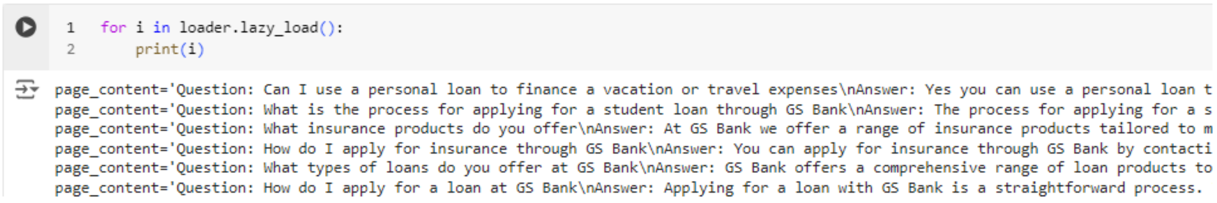
DataLoader expects data in the form of tensors, so we implemented a custom ‘FAQDataset’ class that initializes and converts our tokenized inputs and its corresponding labels to Pytorch tensors.

For training of DistilBERT, RoBERTa and ALBERT model for the task of question classification, we fine-tuned each model over 20 epochs using the Adam optimizer and we set the batch size to 4 and set shuffle to true as we want to shuffle the entire dataset for each epoch. The learning rate was carefully chosen to optimize model performance. The training loop involved calculating the loss and accuracy for each batch and updating the model parameters accordingly.

4.2 Q&A System: Architecture and Implementation

The second half of our project aims to set up a question-answer system using Langchain with Google Palm LLM, FAISS for vector storage and Instructor Embeddings.

We used the same Q&A CSV dataset from GS Bank and converted it to the right LangChain document format using the ‘CSVLoader’ class from the Langchain.document_loaders library.



```
for i in loader.lazy_load():
    print(i)
```

```
page_content='Question: Can I use a personal loan to finance a vacation or travel expenses\nAnswer: Yes you can use a personal loan t
page_content='Question: What is the process for applying for a student loan through GS Bank\nAnswer: The process for applying for a s
page_content='Question: What insurance products do you offer\nAnswer: At GS Bank we offer a range of insurance products tailored to m
page_content='Question: How do I apply for insurance through GS Bank\nAnswer: You can apply for insurance through GS Bank by contacti
page_content='Question: What types of loans do you offer at GS Bank\nAnswer: GS Bank offers a comprehensive range of loan products to
page_content='Question: How do I apply for a loan at GS Bank\nAnswer: Applying for a loan with GS Bank is a straightforward process.
```

Figure 12: Sample data in Langchain document format.

Next, we embedded our document using the Instructor Embeddings from HuggingFace and used FAISS (Facebook AI Similarity Search) library to store the embeddings in a FAISS vector store database.

“A vector store takes care of storing embedded data and performing vector search for us. At the time of query, it embeds the input query and retrieves the embedding vectors that are the most similar to the input query.”- LangChain Documentation

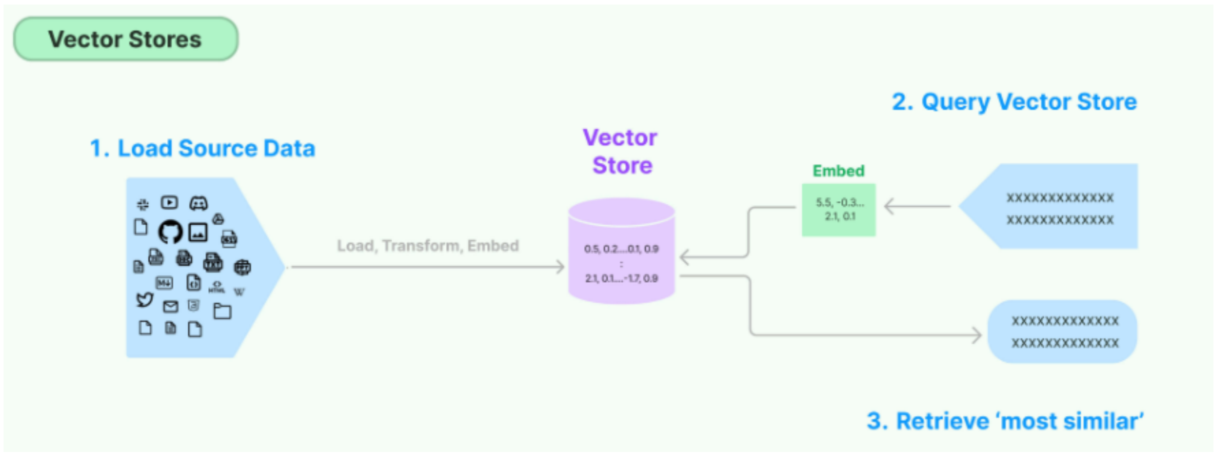
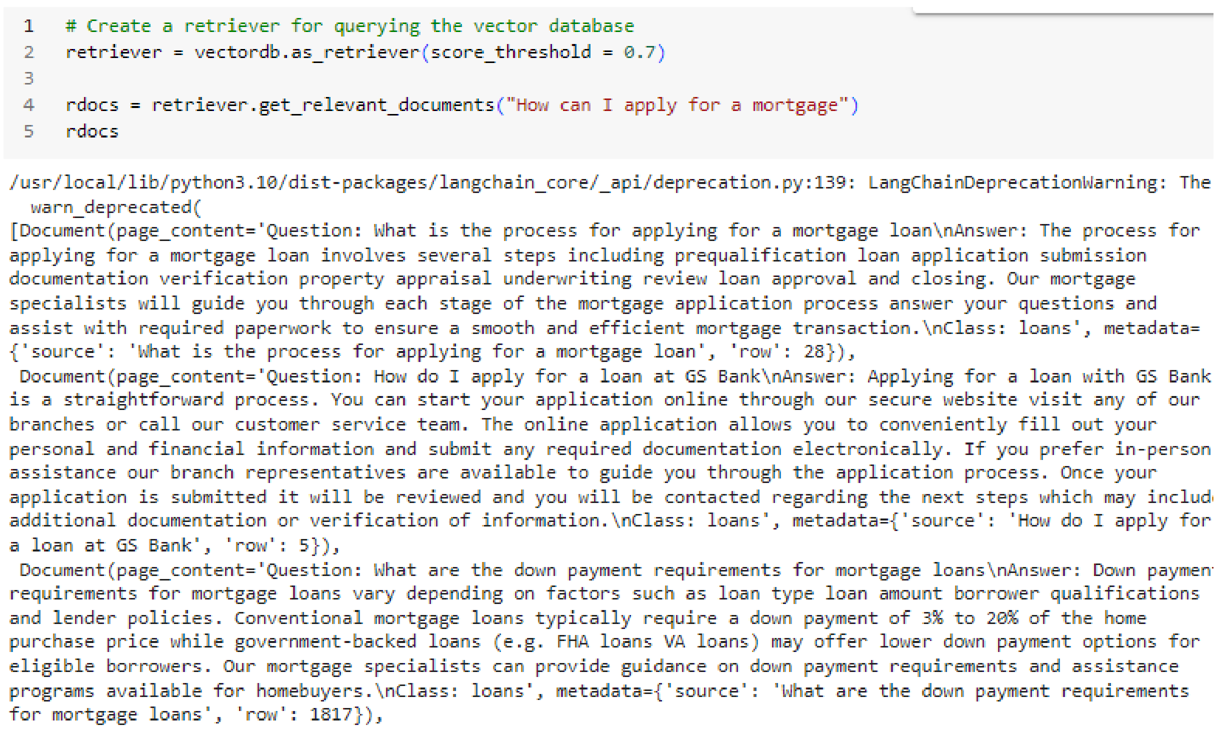


Figure 13: Vector Stores Flowchart (source : LangChain documentation)

The retriever method helps to retrieve similar documents from the vector store and can be parameterized to only return documents with a score above the threshold set.



```
# Create a retriever for querying the vector database
retriever = vectordb.as_retriever(score_threshold = 0.7)

rdocs = retriever.get_relevant_documents("How can I apply for a mortgage")
rdocs
```

```
/usr/local/lib/python3.10/dist-packages/langchain_core/_api/deprecation.py:139: LangChainDeprecationWarning: The
  warn_deprecated(
[Document(page_content='Question: What is the process for applying for a mortgage loan\nAnswer: The process for
applying for a mortgage loan involves several steps including prequalification loan application submission
documentation verification property appraisal underwriting review loan approval and closing. Our mortgage
specialists will guide you through each stage of the mortgage application process answer your questions and
assist with required paperwork to ensure a smooth and efficient mortgage transaction.\nClass: loans', metadata=
{'source': 'What is the process for applying for a mortgage loan', 'row': 28}),
 Document(page_content='Question: How do I apply for a loan at GS Bank\nAnswer: Applying for a loan with GS Bank
is a straightforward process. You can start your application online through our secure website visit any of our
branches or call our customer service team. The online application allows you to conveniently fill out your
personal and financial information and submit any required documentation electronically. If you prefer in-person
assistance our branch representatives are available to guide you through the application process. Once your
application is submitted it will be reviewed and you will be contacted regarding the next steps which may includ
additional documentation or verification of information.\nClass: loans', metadata={'source': 'How do I apply for
a loan at GS Bank', 'row': 5}),
 Document(page_content='Question: What are the down payment requirements for mortgage loans\nAnswer: Down paymen
requirements for mortgage loans vary depending on factors such as loan type loan amount borrower qualifications
and lender policies. Conventional mortgage loans typically require a down payment of 3% to 20% of the home
purchase price while government-backed loans (e.g. FHA loans VA loans) may offer lower down payment options for
eligible borrowers. Our mortgage specialists can provide guidance on down payment requirements and assistance
programs available for homebuyers.\nClass: loans', metadata={'source': 'What are the down payment requirements
for mortgage loans', 'row': 1817}),
```

Figure 14: An example of answers retrieved from the retriever.

To set up a customized Q&A chat system, we also need to generate answers from the relevant documents retrieved from the retriever. We decided to go with PALM large language model from Google as it was free and easy to get an API key.

RetrievalQA chain from LangChain handles the Q&A process. It integrates the LLM with the retriever, passing the input query to the retriever and passing the answers to the LLM to get the context and generate the answers.

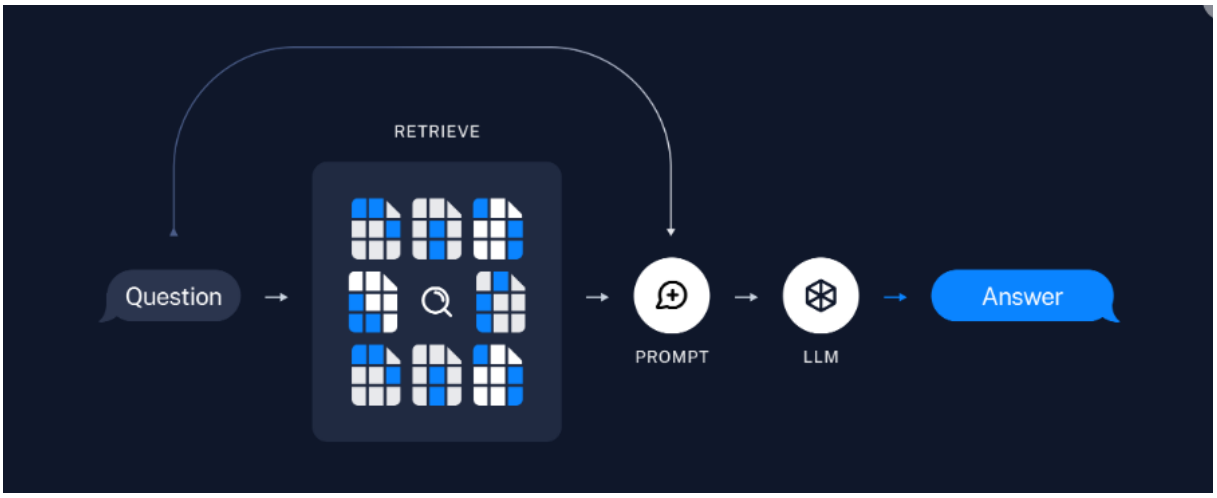
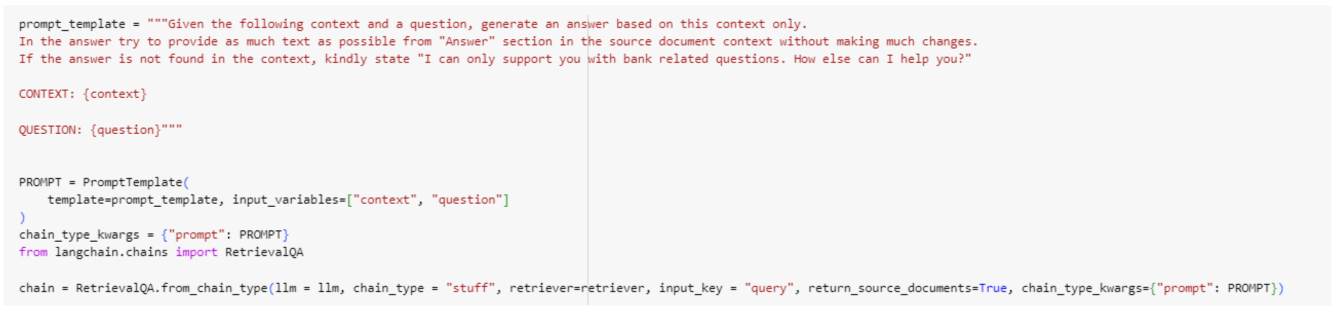


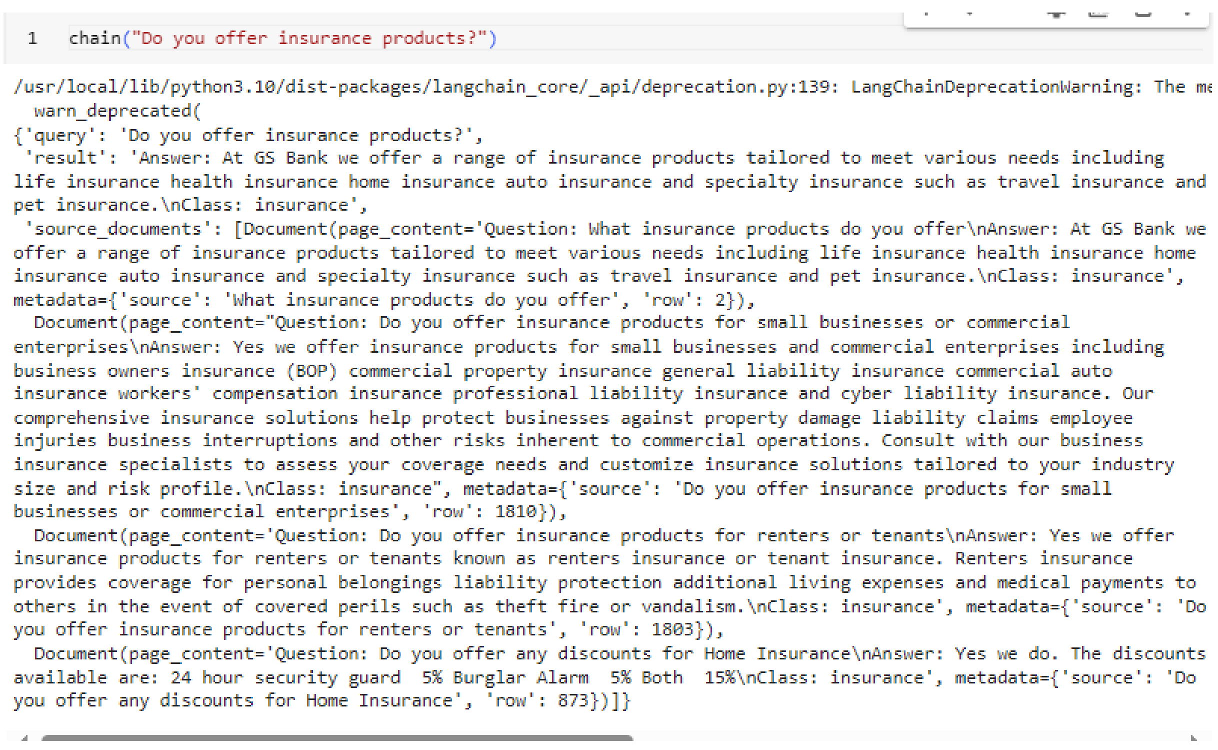
Figure 15: Flow of the Q&A using LangChain and LLM (source: LangChain Documentation)

During our tests, we found that the LLM would often make up answers when it could not find any answers from the retriever. To avoid this, we built a prompt template that includes specific instructions on how to generate answers based on the provided context. If the context does not contain an answer, the LLM responds with our predefined message.



```
prompt_template = """Given the following context and a question, generate an answer based on this context only.
In the answer try to provide as much text as possible from "Answer" section in the source document context without making much changes.
If the answer is not found in the context, kindly state "I can only support you with bank related questions. How else can I help you?"

CONTEXT: {context}

QUESTION: {question}"""


PROMPT = PromptTemplate(
    template=prompt_template, input_variables=["context", "question"]
)
chain_type_kwargs = {"prompt": PROMPT}
from langchain.chains import RetrievalQA

chain = RetrievalQA.from_chain_type(llm = llm, chain_type = "stuff", retriever=retriever, input_key = "query", return_source_documents=True, chain_type_kwargs={"prompt": PROMPT})
```

Figure 16: Code using LangChain and LLM for the Q&A system.



```
chain("Do you offer insurance products?")
```

```
/usr/local/lib/python3.10/dist-packages/langchain_core/_api/deprecation.py:139: LangChainDeprecationWarning: The me
  warn_deprecated(
{'query': 'Do you offer insurance products?',
 'result': 'Answer: At GS Bank we offer a range of insurance products tailored to meet various needs including
life insurance health insurance home insurance auto insurance and specialty insurance such as travel insurance and
pet insurance.\nClass: insurance',
 'source_documents': [Document(page_content='Question: What insurance products do you offer\nAnswer: At GS Bank we
offer a range of insurance products tailored to meet various needs including life insurance health insurance home
insurance auto insurance and specialty insurance such as travel insurance and pet insurance.\nClass: insurance',
metadata={'source': 'What insurance products do you offer', 'row': 2}),
  Document(page_content="Question: Do you offer insurance products for small businesses or commercial
enterprises\nAnswer: Yes we offer insurance products for small businesses and commercial enterprises including
business owners insurance (BOP) commercial property insurance general liability insurance commercial auto
insurance workers' compensation insurance professional liability insurance and cyber liability insurance. Our
comprehensive insurance solutions help protect businesses against property damage liability claims employee
injuries business interruptions and other risks inherent to commercial operations. Consult with our business
insurance specialists to assess your coverage needs and customize insurance solutions tailored to your industry
size and risk profile.\nClass: insurance", metadata={'source': 'Do you offer insurance products for small
businesses or commercial enterprises', 'row': 1810}),
  Document(page_content='Question: Do you offer insurance products for renters or tenants\nAnswer: Yes we offer
insurance products for renters or tenants known as renters insurance or tenant insurance. Renters insurance
provides coverage for personal belongings liability protection additional living expenses and medical payments to
others in the event of covered perils such as theft fire or vandalism.\nClass: insurance', metadata={'source': 'Do
you offer insurance products for renters or tenants', 'row': 1803}),
  Document(page_content='Question: Do you offer any discounts for Home Insurance\nAnswer: Yes we do. The discounts
available are: 24 hour security guard  5% Burglar Alarm  5% Both  15%\nClass: insurance', metadata={'source': 'Do
you offer any discounts for Home Insurance', 'row': 873})]}
```

Figure 17: An example of answers retrieved from the Q&A system

5 Result Discussions

The performance of the DistilBERT, RoBERTa, and ALBERT models was evaluated using metrics such as precision, recall, F1 score, and accuracy. Despite their different architectures and optimization techniques, all three models demonstrated robust performance in classifying customer queries into predefined categories. However, the efficiency and resource utilization varied among them:

- DistilBERT: Demonstrated efficient performance with a significant reduction in computational resources while maintaining high accuracy.
- RoBERTa: Showed superior performance due to its optimized training approach, although it required more computational resources compared to DistilBERT.
- ALBERT: Achieved a balance between performance and resource efficiency, thanks to its parameter-sharing mechanism and improved training objectives.

Weighted Average				
Model	Recall	Precision	F1 Score	Accuracy
DistilBERT	0.93	0.93	0.93	0.93
RoBERTa	0.95	0.95	0.95	0.95
ALBERT	0.93	0.92	0.92	0.92

Table 5: Evaluation metrics for the three model

Roberta outperforms in all the four metrics. The confusion matrices below provide additional insight into the misclassification patterns across different categories:

DistilBERT: DistilBERT's classification performance is noteworthy for its balance between efficiency and accuracy. With an overall accuracy of 93%, DistilBERT demonstrates strong performance across all classes. The precision, recall, and F1-scores are consistently high, indicating robust classification capabilities. However, the confusion matrix reveals minor misclassifications, particularly in classifying instances of Cards and Accounts, where some overlap is evident.
Despite these minor errors, DistilBERT's lightweight architecture and reduced computational requirements make it an excellent choice for this classification task.

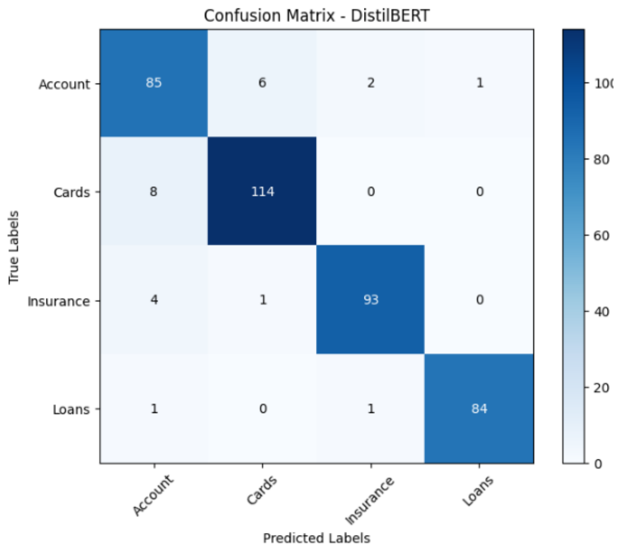


Figure 18: Confusion Matrix for DistilBERT

RoBERTa: RoBERTa outperforms DistilBERT and ALBERT in terms of overall accuracy, achieving an impressive 95%. This superior performance is attributed to RoBERTa's optimized training regimen, which includes larger mini-batches, extended training periods, and the removal of the next sentence prediction objective. The classification report indicates exceptional precision and recall values, with F1-scores nearing perfection. The confusion matrix confirms RoBERTa's efficacy in minimizing misclassifications, although minor errors remain.

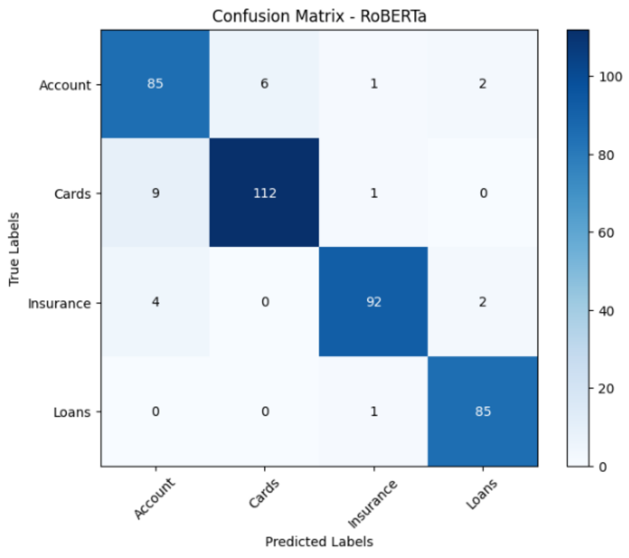


Figure 19: Confusion Matrix for RoBERTa

ALBERT: ALBERT provides a balanced approach, with an overall accuracy of 92%. Its parameter-sharing mechanism and the use of sentence order prediction contribute to a more efficient model without significant performance trade-offs. The classification report shows strong precision and recall , with F1-scores reflecting high classification quality. The confusion matrix indicates occasional misclassifications, primarily between Accounts and Cards. ALBERT's ability to maintain high accuracy with reduced resource consumption underscores its suitability for applications requiring efficient and scalable solutions.

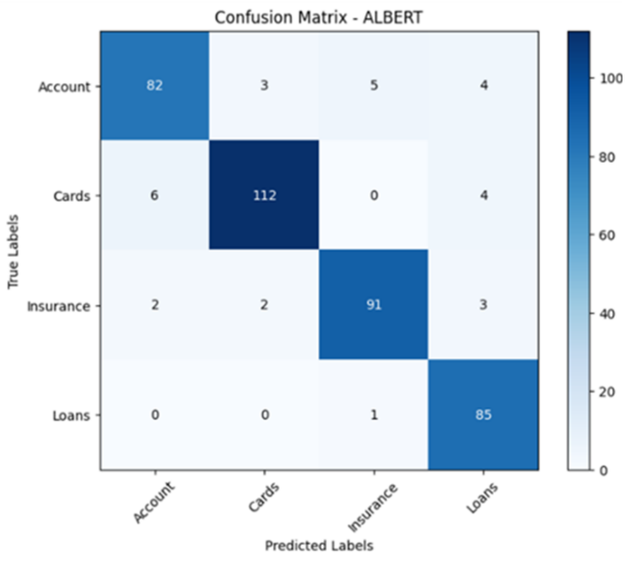


Figure 20: Confusion Matrix for ALBERT

Across all three models, RoBERTa stands out with the highest accuracy and F1-scores, particularly excelling in precision and recall. DistilBERT, while slightly behind in overall accuracy, offers a commendable balance of performance and efficiency, making it highly practical for resource-constrained environments. ALBERT, meanwhile, achieves a middle ground, combining efficiency with robust performance metrics.

The confusion matrices across all models reveal a common challenge in distinguishing between certain classes, such as e.g., Accounts and e.g., Cards. These misclassifications suggest areas for further refinement, potentially through increasing the volume of training data or additional contextual data to improve class separability.

In conclusion, each model presents unique strengths, with RoBERTa offering the highest accuracy, DistilBERT providing an optimal balance of performance and efficiency, and ALBERT delivering a cost-effective yet robust solution. The findings from this evaluation underscore the potential of transformer-based models in advancing the state-of-the-art in question classification tasks.

6 Conclusion and Outlook

In conclusion, our work demonstrates the feasibility and effectiveness of using pre-trained models such as DistilBERT, RoBERTa and ALBERT for question classification in the domain of customer service within the banking and finance sector. The results suggest that leveraging pre-trained language representations can significantly enhance the performance of question classification systems in banking and finance. By employing models such as DistilBERT, RoBERTa, and ALBERT, we were able to systematically classify customer inquiries into distinct categories, enhancing the efficiency and accuracy of response mechanisms at GS Bank. This categorization not only streamlines support workflows but also provides valuable insights into customer preferences and needs, thereby aiding in the optimization of the bank's product offerings. Our comparative analysis of the three models revealed that RoBERTa consistently outperformed the other models in terms of precision, recall, F1 score, and accuracy. This highlights the robustness and high performance of RoBERTa for question classification tasks. However, it is essential to note that each model possesses unique strengths: DistilBERT offers faster processing and efficiency, making it suitable for resource-constrained environments, while ALBERT balances performance with memory efficiency.

The integration of a retrieval-based Question Answering (QA) system utilizing a Large Language Model (LLM) further underscores the potential of combining machine learning models with practical applications to improve customer service experiences significantly. Our implementation using LangChain, FAISS, and Google's PaLM LLM showcased the ability to provide accurate and timely responses to customer queries, enhancing the overall service quality.

Moving forward there are several avenues for further research and development. Future work could explore:

- Enhanced Fine-Tuning: Investigating additional fine-tuning strategies and hyperparameter optimization to further improve model accuracy and robustness.
- Domain-Specific Data: Expanding the dataset to include more diverse and domain-specific queries, which could improve the model's adaptability and performance in real-world scenarios.
- Real-Time Implementation: Developing real-time systems that can seamlessly integrate with the bank's existing infrastructure to provide instant responses to customer inquiries.

- User Feedback Loop: Implementing a feedback loop where customer interactions can continuously refine and improve the model's performance over time.

7 Acknowledgments

We would like to express our deepest gratitude to the following individuals and organizations for their invaluable contributions to this project:

- HuggingFace: For providing the pre-trained transformer models and extensive documentation that greatly facilitated our model implementation and fine-tuning processes.
- PyTorch Community: For developing the deep learning framework that served as the backbone of our project. The flexibility and efficiency of PyTorch enabled us to train and optimize our neural networks effectively.
- LangChain Developers: For creating the robust framework that allowed us to efficiently build our QA system using LLMs.
- FAISS Developers: For the powerful similarity search library that enabled efficient vector storage and retrieval operations in our project.

Lastly, we extend our sincere appreciation to our Professors of the Institute of Mathematics at the University of Bern for their continuous support and guidance throughout our Certification of Advanced Study Modules. Special thanks to our advisors and peers who provided invaluable feedback and encouragement.

List of Figures

Figure 1: Data distribution.

Figure 3: Calculate number of words

Figure 4: Histogram showing the number of words in each question.

Figure 5: Violin plot to show the distribution of words within a class.

Figure 6: Histogram to show the distribution of characters in each question.

Figure 7: Chart showing the most common words in our dataset.

Figure 8: Word Cloud showing the most common words.

Figure 9: Chart showing most common words after data preprocessing.

Figure 10: Shuffled sample data with only the necessary columns.

Figure 11: Code tokenizing our train and test data for DistilBERT.

Figure 12: Sample data in Langchain document format.

Figure 13: Vector Stores Flowchart (source : LangChain documentation)

Figure 14: An example of answers retrieved from the retriever.

Figure 15: Flow of the Q&A using LangChain and LLM (source: LangChain Documentation)

Figure 16: Code using LangChain and LLM for the Q&A system.

Figure 17: An example of answers retrieved from the Q&A system

Figure 18: Confusion Matrix for DistilBERT

Figure 19: Confusion Matrix for RoBERTa

Figure 20: Confusion Matrix for ALBERT

List of Tables

Table 1: Sample data

Table 2: Data columns and data type

Table 3: Sample data after text preprocessing and label encoding.

Table 4: Train and Test data split

Table 5: Evaluation metrics for the three model

REFERENCES

1) [1910.01108] DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (arxiv.org)
2) https://huggingface.co/docs/transformers/en/model_doc/distilbert
3) The DistilBERT model architecture and components. | Download Scientific Diagram (researchgate.net)
4) [1909.11942] ALBERT: A lite BERT for Self-Supervised learning of Language Representation (arxiv.org)
5) [1907.11692] RoBERTa: A Robustly Optimized BERT Pretraining Approach (arxiv.org)
6) A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification (wiley.com)
7) Welcome to Faiss Documentation — Faiss documentation
8) PaLM - Wikipedia
9) Introduction | 🦜🔗 LangChain
10) Tutorials | 🦜🛠️ LangSmith (langchain.com)
11) LangChain 0.2.5 — LangChain_api_reference-module-langchain.retrievers