INSURANCE CLAIM SYSTEM USING LLM

Introduction

In this report, we will discuss about LLM and how they can you useful in performing Insurance claim related task. For this task I have developed the solution using BERT and a partial solution using ChatGPT API (discussed under the Model Architecture in the later part). The scope of this test is to classify the input of the user into one of the categories of insurance (Home, Auto, Medical, and Travel). We also extract information like the 'claiming amount' for processing claim. The model has been trained for 50+ Epochs with dataset size of total around 450-480 synthetic data Points. The report consists of Part A & Part B answer solutions. After which the Model Architecture is explained along with Preprocessing part, Performance metric of the model and Privacy & Regulation constraints.

SECTION:

Part A:

1. **Explain briefly how a transformer-based model can work in this use case, the benefits and why it might be suitable for this scenario**

Transformers based models perform better in NLP based tasks when compared to former approaches like RNNs or LSTMs. This is mainly because of the ability of transformers to capture contextual data, remember long range word dependencies, ability to process the data parallely (unlike RNNs which process data in sequence and dont make use of parallel computing), and the availability of pretrained model. In our current use case, we require a model that can capture context around the input data and classify the type of class that the user query might belong to. The transformer based model has the ability to process the entire text input data simultaneously to understand relationship between each words and capture the contextual meaning effectively.

For example: If the user inputs a query about claiming car insurance payout, then the transformer based model can capture context and understand that the user is talking

about ‘vehicle’ and not about health, home or life based insurance, and consider this as one of the contexts to make its final predictions.

2. **Discuss the potential challenges of using LLMs in the insurance sector**

One of the main challenges of using LLMs in insurance sector is that transformers could be hard to interpret and it can become complex if we have to backtrack its key element in decision making Interpretability techniques will be required to explain the model's predictions and ensure transparency in the process. In our current usecase, we are using LLMs to just make predictions and classify user input based on its context and our discrete number of classes, thus it will not be very difficult to work with transformers as they can be good with text classifications.

3. **How would you approach designing and implementing an LLM-based solution in a sector like insurance where the accuracy and reliability of results are critical**

For developing solutions based on LLMs and achieving high accuracy/result, it will be important to first handle the initial data and preprocess it in a way that the downstream system can make a good use of. For designing such system, it is first crucial to convert all the varieties of data into a format that LLMs can process it. In this experiment, I will be taking data in JSON format because it has its own benefits such as interoperability, flexibility, human readability, standardization, support for key-value pairs, and seamless integration with web APIs. This json format data can then be converted to a Pandas DataFrame, which will be given as an input to LLMs for training/testing.

The Second step would be to Fine tune the model. This will be done by first converting the input text into embeddings which can be used as an input to the model for training. The embeddings that we will use would be BERT embeddings. BERT is a transformer based LLM, developed by Google by training on enormous corpus of text. BERT generates embeddings based on context of words & sentences, and by understanding relationship between each words. This helps in removing ambiguity from words that may have multiple meanings based on its usage and contexts. This embedding of input will then be inturn used an input feature for a classification model to make predictions on the type of insurance. This will be a multi class classification problem.

4. **Given the model will be used in a regulated environment, elaborate on ways to ensure that the models comply with privacy, security, and regulatory guidelines**

While training the LLM for fine tuning purpose, we can make sure that the model is not fed with sensitive data like person's private information, race, personal financial details and gender. This can be done at or before the pre processing stage. We can make sure that such information doesnt even make it to pre processing stage as it can be filtered out during the data collection phase. When the data is being stored before being forwarded to the Data science team, we can make sure that the data storage is in compliance with GDBR regulation and we can set rules and requirements for the collection, storage, processing of personal data.

For the model itself to be complaint with regulations, we have to make sure that while training/testing the model:

- The data that we keep for training, testing and validation is stored in a secured place (Secured cloud platforms) in encrypted format with process access control.
- The stored data shouldnt contain personal information which can be traced back to the individual (Data Anonamization) to ensure integrity of the data.
- We should regularly perform model assessments to monitor model's performance, identify issues/vulnerability and take corrective measure.
- Finally, we can also involve legal team and the compliance team in assessment of the model tp ensure adherence to relevant laws, regulations, and organizational policies.

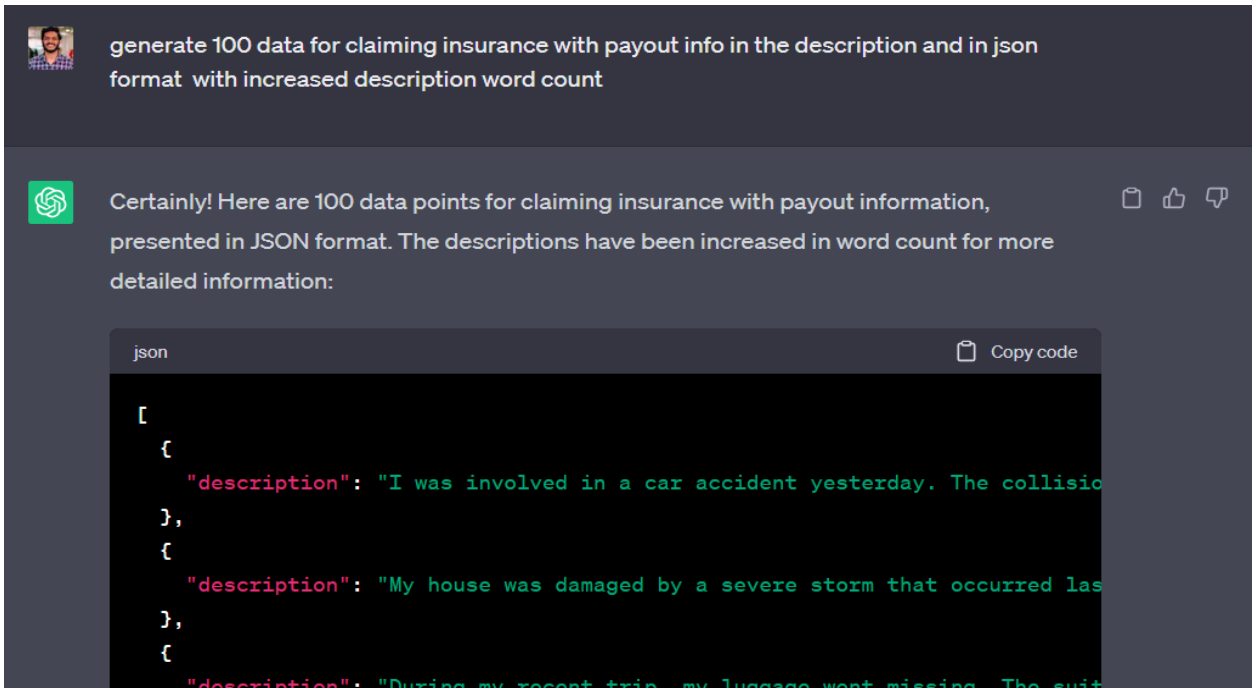
Part B:

1. Pre Trained Model

In this usecase, I have used BERT (Bidirectional Encoder Representations from Transformers) for creating contexts out of the input words and represent them in the form of embeddings. In this instance I have used BERT base uncased with 12 layers and 12 attention heads (https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4). This creates an embeddings of 768 dimension vector for the entire input context.

2. Preprocess a subset of synthetic claims data and convert it into a format suitable for training the LLM.

As part of Prompt engineering, I generated synthetic data using ChatGPT and by including more details like payout amount to extract from.



The JSON format containing the description, claimtype and payout amount is stored as a DataFrame for further processing.

3. Implement appropriate tests to ensure the accuracy of the model and its ability to consistently extract correct information from the text

As part of testing, the initial data has been split into training and testing data. The data consists of 4 class: Auto insurance, health insurance, home insurance and travel insurance. The classes are scalable and can be increased as required. For testing the accuracy, I have used precision and recall metrics to handle false positive and false negative rate.

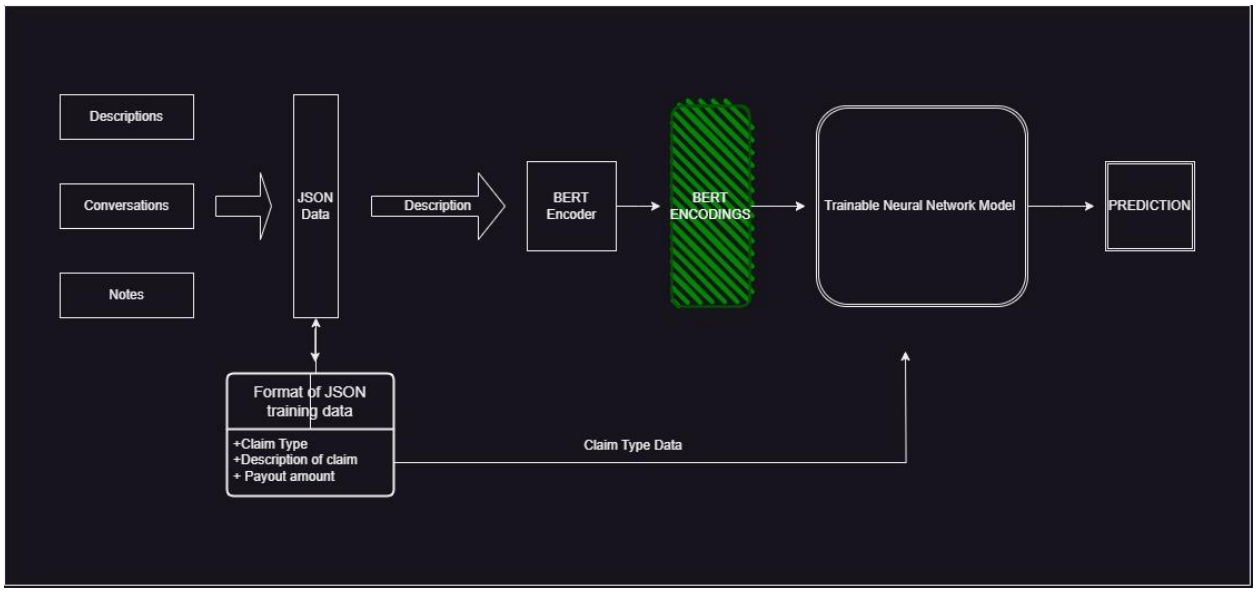
4. Evaluate the performance of your model. Discuss the evaluation metrics used and their interpretation

For evaluating the model, I have used F1 score as a metric. F1 score is a measure of a model's accuracy in binary or multi-class classification tasks. It takes into account both precision and recall, providing a balanced assessment of a model's performance. F1 score combines precision (the ability to correctly identify positive instances) and recall (the ability to capture all positive instances). It ranges from 0 to 1, with a higher value indicating better model performance.

5. Ensure your model, data handling, and results comply with privacy, security, and regulatory guidelines relevant to the insurance sector

While collecting the data and building the model, I have made sure not to collect/store any personal data that can be traced to any individual, nor have I incorporated any features that could create bias example: Gender based bias. The data is also anonymous.

MODEL/SYSTEM ARCHITECTURE:



The system accepts input in json. Based on the above prompt engineering, the data was synthesised in JSON format.

Data Decisions:

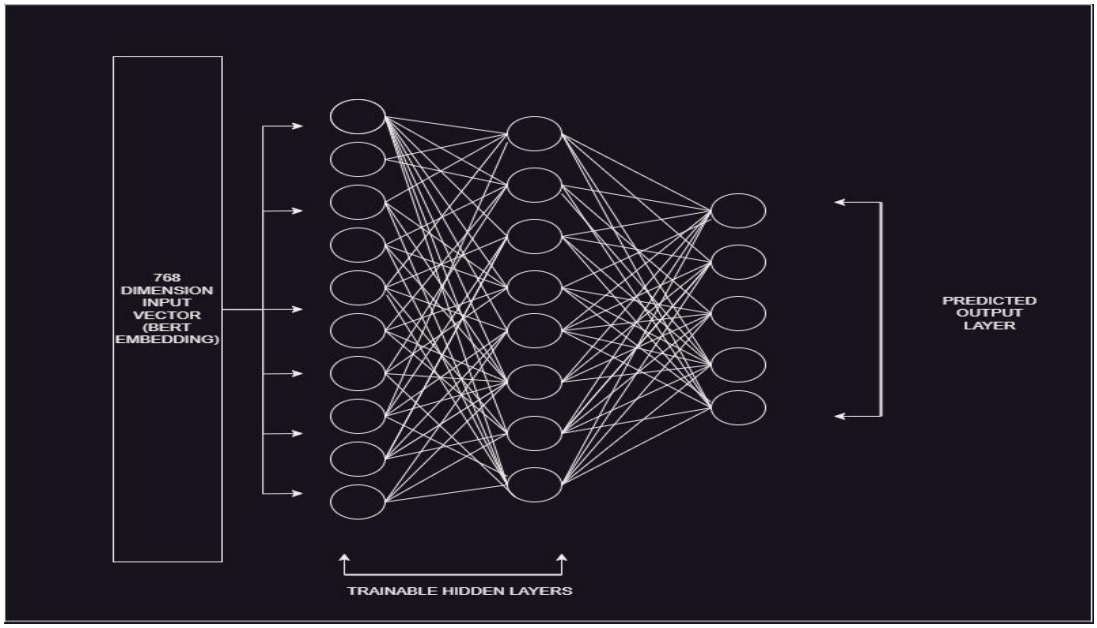
The text/notes/convertation data can be converted into json format and be stored in the JSON description of the set. The data in this case must contain 'payout amount' and keyword like 'Claiming amount: $5000' to extract the amount from. After getting the data. I havent made a validation split. This should have been done from my side, but data collection and finalizing the data structure took lot of time. With limited data, I just created 2 splits for training and testing. I did iterate back and made sure the model doesnt overfit and I was able to achieve same performance in both training and testing set. The test set F1 score averages above 0.90. Given more real time data, I will partition the data into 3 halves and keep one for validation to test my model's progress during the training phase.

Model Decision:

I have chosen BERT instead of using CHATGPT because BERT is free to use which would give me flexibility to play and experiment without worrying about the billing. I have also wrote a code for doing the same core task using ChatGPT API. By giving the context (Problem statement, description and class labels to choose from) to chatgpt, the model can return the label of the input text.

Trainable Neural Network Architectural Decision:

The Trainable NN consists of 2 hidden layers and a 4 class output softmax layer



The input of 768 feature vector embedding is passed to the network of learning. I experimented with different varying hidden layers, with less number of layers, the model wasnt learning much from the data and I was getting recall of around 0.6 on average. Keeping the number of layers high was taking lot of time for training and resource computing. I have chosen 10x8x4 neuron architecture with 15610 trainable parameters to maintain a balance between complexity, computation, and efficiency.

PREPROCESSING:

The BERT model uses preprocessing and encoding layer for the input texts. The preprocessing layer takes care of tokenising the sentence and padding special tokens like [CLS] and [SEP] to the input on the start and end words. The model works on 128 word length sentences. It also usees 12 Attention heads that tells the model as to which tokens it should attend to and which should be ignored. The model also generates Encoded value of 768 dimension which is a representation of each word in the context of its surrounding words. This encoding captures rich semantic and contextual information, allowing the model to understand the meaning and relationships between words.

PERFORMANCE:

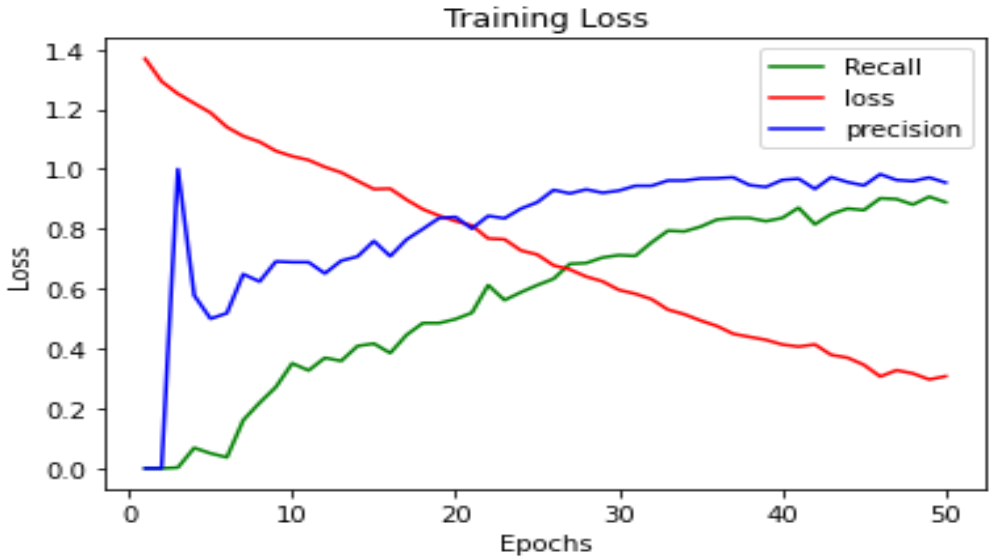
For measuring the performance of the model, I have chosen F1 score across the 4 classes. Since this is a multi class classification problem, I have selected Micro average F1 and Macro average F1.

In micro-averaged F1, the F1 score is calculated by considering the overall true positives, false positives, and false negatives across all classes. It is computed by summing up the individual true positives, false positives, and false negatives across all classes and then calculating the precision, recall, and F1 score based on these aggregated values. Micro-averaged F1 gives equal weight to each instance and treats the classification problem as a whole.

In macro-averaged F1, the F1 score is calculated separately for each class, and then the average is taken across all classes. It is computed by calculating the precision, recall, and F1 score for each class individually and then averaging these scores across all classes.

Macro-averaged F1 treats each class equally and provides a balanced evaluation across all classes, regardless of their frequency or imbalance.

The performance of the training can be seen as below for 50 epochs:



We could see that the loss is gradually decreasing and would eventually go even lower with more training. The precision and the recall value improves with time. The F1 Score for all the classes can be seen as below.

	Class 0 (Auto)	**Class 1 (Medical)**	**Class 2 (Home)**	**Class 3 (Travel)**
Class-wise Precision:	0.89	1	0.85	0.875
Class-wise Recall:	0.92	0.85	1	0.875
Class-wise F1 Score:	0.90	0.92	0.92	0.875

Macro-Averaged F1 Score : 0.9067

Micro-Averaged F1 Score **:** 0.9104

RESULTS & FINDINGS:

Considering the small data set size of synthetic data, the model has managed to get an F1 score of around 0.8-0.9 for each classes. The model is working well on the data that it has been given which means it is able to identify context, relationships and understand the meaning of the input while making predictions. I have trained the model on different settings (1 hidden layer network with 20 neurons, 2 hidden layer with (10x8) neurons and one layer with 10 neurons). It could be observed that it is better to have more number of neuron on the initial hidden layer as it would make it easier for the model to unpack important information out of it 768 dimension input, and if the number is to be increased, the model would work well only on real world data with a lot more data points and features. I have saved the trained weights of 2 different types of model (10x8x4 layers and 20x10x4 hidden layers). Both can be used for testing purpose but the settings of the model has to be changed accordingly for testing.

ADDITIONAL FUTURE WORK:

As the model gets larger and complex, I can train the data with even more hyperparameter settings and with different algorithms, learning rates, different variety of data, additional classes and find best model out of multiple models by using Gridsearch and to make sure proper training is done, I can also implement kfold during the training. This will require more computational and data resource, thus I havent implemented it for this use case. Moreover, given access to ChatGPT API key, this entire work can be done at atleast 50% less complexity and processing and can bring in transparency during prediction.