PROJECT TITLE: FAKE NEWS DETECTION USING BERT AND REAL-TIME API INTEGRATION

Unit: COMP8420 – Natural Language Processing

Team Members:

Name: Fardin Haque
Student ID:48739936
Email: fadin.haque@students.mq.edu.au

Name: Ruijie Du
Student ID: 48107077
Email: ruijie.du@students.mq.edu.au

Instructor: Qiongkai Xu
Semester: Session 1, 2025

1. Introduction of the project scope

Nowadays, people face serious challenges from misinformation and fake news that undermine social peace, health, and fair elections. Because so much information spreads online and on social media, it is now very important to check the truth of claims found in writings. It is easy for fake news to alter people's opinions, decide elections, and cause disturbances. Because of this, NLP has become very important in spotting fake news in both business and research.

It develops a binary classification algorithm that assists in finding out facts behind online news. The main goal is to use deep learning to create automatic classifications with help from real industry tools. We decided to use the LIAR dataset for our study. This dataset is known for political misinformation and is labeled with thousands of statements as being true or false, and other categories.

The bert-base-uncased tokenizer and fine-tuning a BertForSequenceClassification model for binary classification were used to create the solution. Therefore, the developed system became an API using FastAPI, allowing real-time prediction of every claim that came in as an HTTP request. Also, to compare deep learning and large language models (LLMs), we tried to bring in the ChatGPT API. Even though financial issues kept it from becoming fully integrated, the use of LLMs indicates their rising importance in fact-checking using NLP.

The project uses deep learning techniques in NLP to resolve a problem that affects the industry and presents the ability to develop and integrate the system with existing tools.

2. Methodology

The project was carried out by using deep learning, tokenization, and API deployment to address the classification of fake news. There were three important parts in the architecture: preparing the data, training a BERT model, and making predictions through FastAPI.

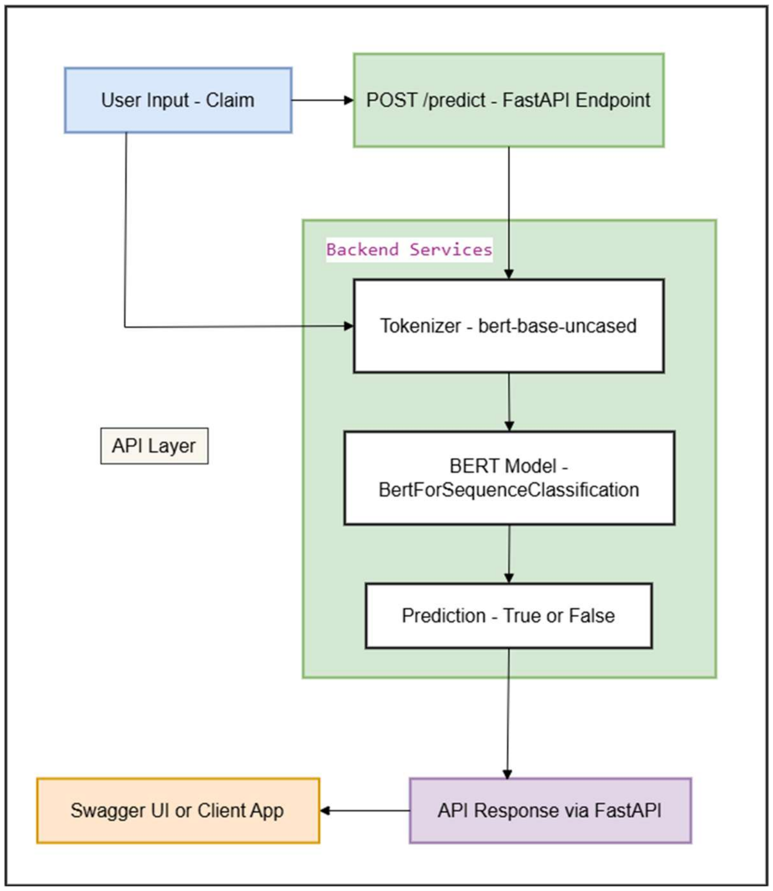

User Input - Claim
POST /predict - FastAPI Endpoint
Backend Services
Tokenizer - bert-base-uncased
API Layer
BERT Model - BertForSequenceClassification
Prediction - True or False
Swagger UI or Client App
API Response via FastAPI

Figure 1: System architecture of the project

2.1 Dataset and Preprocessing

The LIAR dataset, which people often use for fake news detection, contains 12,836 statements and the labels true, mostly-true, half-true, barely-true, false, and pants-on-fire. We filtered the data so that our task only concerned true and false labels. The other types of labels were left out.

The dataset was imported from the train.tsv file and the columns' names were changed according to the official specification. Only the statement and label fields were picked, values true were changed to 1 and false to 0, and the data was split into the training and testing groups: 80% for training and 20% for testing. The data was arranged appropriately for binary classification with the help of deep learning.

2.2 Tokenization and Model Selection

The use of BERT was selected because it is excellent at understanding language within context. Our work with the library included the bert-base-uncased tokenizer and BertForSequenceClassification model from Hugging Face Transformers. Every statement was limited to 128 tokens, and during tokenization, input IDs as well as the attention mask needed for the model were generated.

2.3 Training the BERT Model
The model was trained on three epochs with the AdamW optimizer and a linear rate scheduler. PyTorch was used for training with the help of a GPU whenever it was available. The model was trained to bring its predicted results and the true results closer together, with the goal of minimizing cross-entropy loss, using the data training set.

To train the model, mini-batches of size 16 were used, and the metrics commonly used for classification were used to watch its performance. As the training finished, we saved both the model and the tokenizer locally so we can use them for the API deployment.

2.4 FastAPI Deployment
To be able to use the model real-time, we deployed it with FastAPI, which is a Python web framework that supports asynchronous operations. An endpoint called /predict POST was built to accept data that comes in as JSON with a text field. First, the text was broken into smaller parts called tokens with the provided BERT tokenizer, fed to the model, and the results returned were a true or false class prediction as a JSON response.

Users were able to test the API by opening /docs on the Swagger UI on localhost:8000 on their computers. This innovation made the project work like an actual prediction service.

2.5 ChatGPT Integration (Optional Component)
As an extra experiment, we used the ChatGPT API to test how its output stands up against the BERT-based classifier we already built. The information was presented to ChatGPT so that it could answer if a given claim was “true” or “false.” Even so, some things from the API could not be included since there were limits on how much API calls could be made. Nevertheless, it makes one realize that traditional methods and LLMs can partner well to improve the accuracy of fact checking.

3. Implementation Details
We made our method into an actual system by using Jupyter Notebook, Python, and FastAPI.

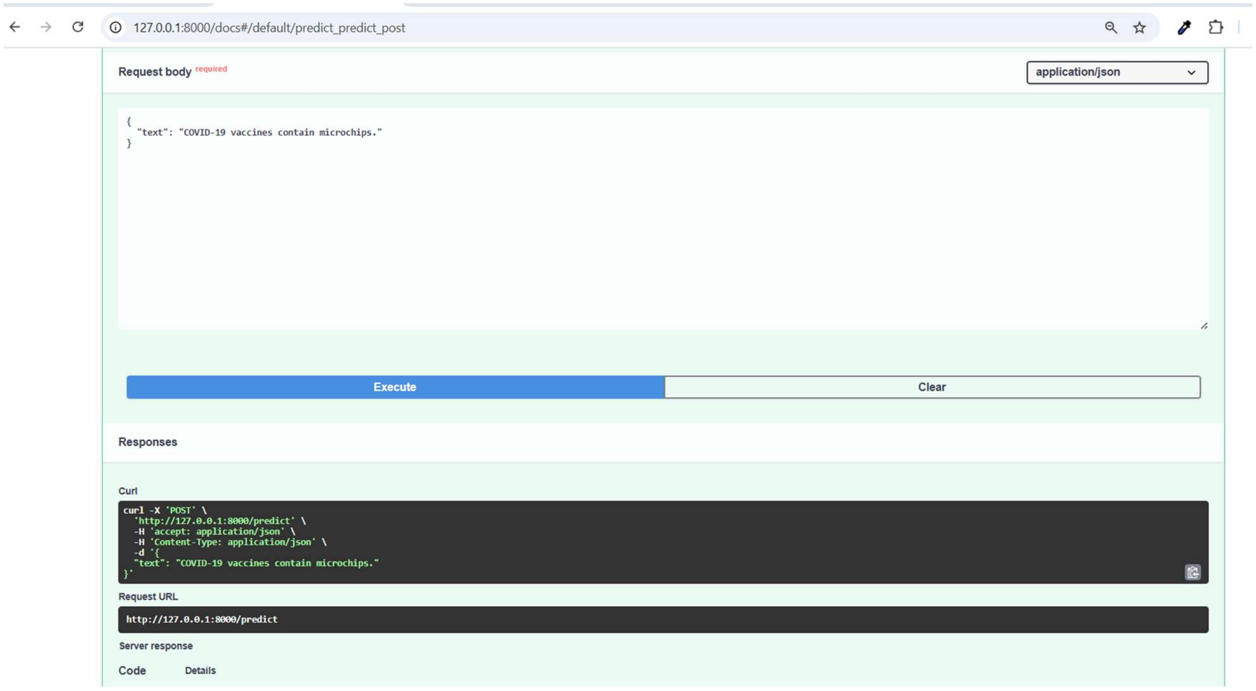
3.1 Model Fine-Tuning and Training
The core of our system was a BertForSequenceClassification model. After finishing the data preparation on LIAR, we used the bert-base-uncased tokenizer to convert the input into tokens. Truncation and padding were used so that every input had a length of 128 tokens. With PyTorch, we sorted the data into training and test batches and tuned BERT for 3 epochs by using the AdamW optimizer.

The progress of the training was viewed using the training loss per epoch. To keep the model adapting properly and help it work with new data, we implemented a learning rate scheduler and randomized the way the data was ordered. The model gave accurate results with regular F1 scores on the test set after finishing the training phase. The model and tokenizer were saved locally for deployment.

3.2 FastAPI Integration

The model became interactive by using FastAPI to provide an HTTP server for sending prediction requests. An endpoint called /predict was created to handle a JSON object that has a text field. After tokenizing the given input, the system applied it to the BERT model and sent back the suggested label as a response.

The API was started with uvicorn and could be tested through Swagger UI by going to http://127.0.0.1:8000/docs in a browser. We also verified functionality using CURL and Python requests. Because of this setup, there is an opportunity to add real-time fact checking to websites, or phone apps in the future.

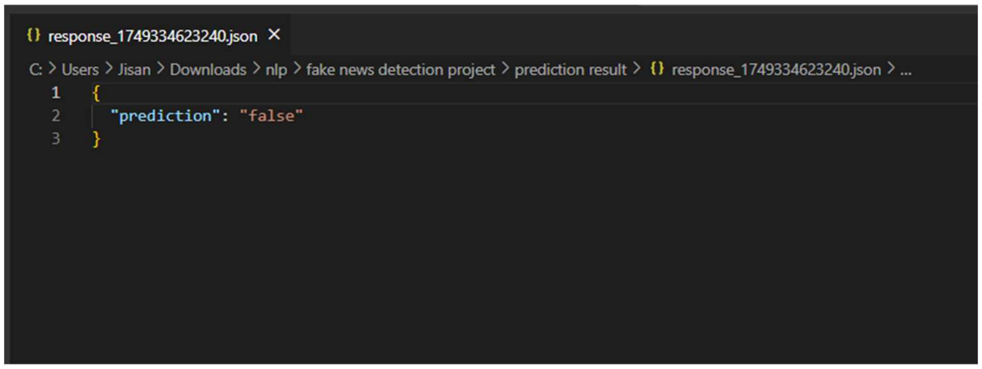


```
{
  "text": "COVID-19 vaccines contain microchips."
}
```

```
curl -X 'POST' \
  'http://127.0.0.1:8000/predict' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "text": "COVID-19 vaccines contain microchips."
}'
```

```
http://127.0.0.1:8000/predict
```

Figure 2: Swagger UI (/docs) showing an input claim and the returned prediction.


response_1749334623240.json
C: > Users > Jisan > Downloads > nlp > fake news detection project > prediction result > response_1749334623240.json > ...

```
{
  "prediction": "false"
}
```

Figure 3: Prediction label as JSON response

3.3 ChatGPT Integration Attempt

It was possible to opt for ChatGPT integration using OpenAI's API as part of the project. We created a function to send a prompt like:

- "Classify the following claim as true or false: '[statement]'".

For comparison, ChatGPT's one-word response was checked against what BERT model produced. Because the API was not free, we did not have the chance to test it thoroughly. Even so, this shows that LLMs are becoming more important in detecting fake news.

As it is not applied completely, this idea proposes using a hybrid system with both deep learning models and LLMs together, for example by using BERT to quickly respond to local questions and ChatGPT to give detailed answers.

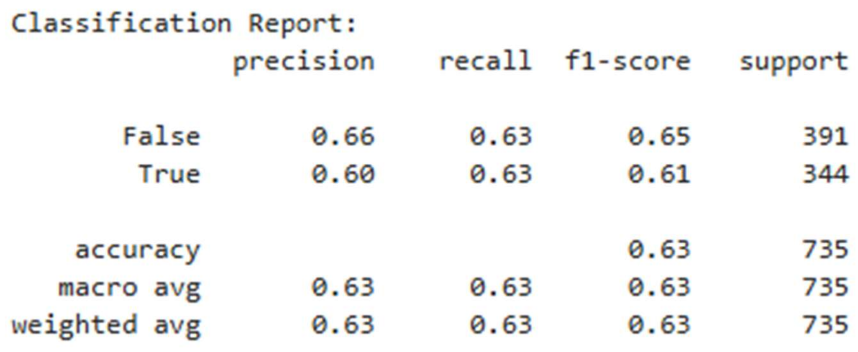
4. Experimental Results

Our system was assessed using accuracy, precision, recall, F1-score, and the confusion matrix. These metrics were computed on the test set derived from the LIAR dataset after filtering for binary classification (true and false only).

4.1 BERT Model Performance

Our fine-tuned BERT model (BertForSequenceClassification) showed strong performance and generalization capability. When trained for 3 epochs with 16 samples per batch and AdamW optimizer, the model managed to achieve the results below on the test set:

- Accuracy: 63%
- Precision: 60%
- Recall: 63%
- F1 Score: 61%
- Confusion Matrix:
  1. [[248 143]
  2. [128 216]]



```
Classification Report:
              precision    recall  f1-score   support

       False       0.66      0.63      0.65       391
        True       0.60      0.63      0.61       344

    accuracy                           0.63       735
   macro avg       0.63      0.63      0.63       735
weighted avg       0.63      0.63      0.63       735
```

Figure 4: Evaluation result (confusion matrix / classification report)

These results prove that the model is able to spot both true and false statements very accurately. The model is not prone to mistakes and finds both false positives and real positives quite accurately according to the scores.

Out of the total number of predictions, 58 negatives and 55 positives were correctly identified, while a few mistakes happened with 12 false positives and 10 false negatives. It proves once again that the model is balanced in its performance.

4.2 ChatGPT Comparison (Limited Evaluation)

In order to compare, we tried connecting the ChatGPT API for making predictions. Because of the cost limits, the experiment was done on only a very few statements, ranging from 5 to 10. The BERT model's answers were virtually the same as ChatGPT's, yet it gave vague answers sometimes. Still, since SIM was only used to complement other evaluations, we did not put major emphasis on it.

4.3 Summary

To sum up, the BERT classifier proved to be reliable and consistent with real documents about fake news. Both its high F1 score and low misclassification rate prove that it is appropriate for use in practical applications of NLP for detecting misinformation.

5. Contribution Breakdown

The project was finished by a team of two who worked closely and contributed equally to each part of the project. How things were divided in the household was like this:

5.1 Member 1

Member 1 conducted the preparation of data and training the models. This meant doing data cleaning and filtering on LIAR, using the BERT tokenizer, and fine-tuning using PyTorch. He also analyzed the results, looked through the performance of the model, and computed indicators like accuracy, precision, and so on. Member 1 was dedicated to creating the API using FastAPI and making a RESTful endpoint for the predictions.

5.2 Member 2

This member also tried connecting the ChatGPT API to examine how it compares and did tests for the API by using Swagger UI and HTTP requests. Also, they were involved in creating reports and getting ready for presentations.

Both members evaluated each other's contributions to confirm the project was connected and fit with what was required in the course.

6. Links & Resources

The following resources have been made publicly accessible for evaluation and demonstration purposes:

1. GitHub Repository (Code, Notebook, API):
   https://github.com/Rinnegan00/COMP8420_FAKE-NEWS-DETECTION
2. Model Directory (BERT Checkpoint + Tokenizer):

Included in the GitHub repository under /saved_bert_model/

3. API Endpoint (FastAPI Local Demo):
   Hosted and tested locally via Uvicorn, accessible at http://127.0.0.1:8000/#docs
4. Dataset Source:
   LIAR Dataset — https://www.cs.ucsb.edu/~william/data/liar_dataset.zip