LLM ON ENERGY POLICY

John Beall, Javier Celestino, Rena Nakada, Alexander Park, Kevin Sheard, Michelle Wan

INTRODUCTION

In the age of large language models like ChatGPT and LLaMa, data is king. Companies and researchers invest large amounts of human resources in collecting all the data they can. However, before organizations can use this valuable data, it must first be organized in a structured way. Text classification allows organizations to efficiently and cost-effectively organize any type of text, such as e-mails, legal papers, databases, and other documents. AI tools like BERT exist that can help in automating this task. BERT stands for Bidirectional Encoder Representations from Transformers and utilizes transfer learning to perform language processing tasks. We combined the web scraping library BeautifulSoup and BERT for this project to create a binary classification model. The data we used for this project was collected from PetroWiki & AAPG Wiki.

METHODS

In the first step of our project, we had to ensure scraping was allowed for each site by locating the root of the website for each wiki site. Once we had confirmation of the presence of a robots.txt file in each wiki site, we chose to do a binary classification to categorize data into one or more classes. This classification would be done by scraping multiple sources from two different sites: PetroWiki and AAPG wiki. We planned our scraping tool by using the popular Python library BeautifulSoup, which specializes in web scraping by making extracting data from HTML and XML files easier. To implement the scraper, we selected only the necessary body of paragraph within the chosen link for each wiki site and began to extract .txt files for our pre-build LLM model Bert to analyze. The scraping process extracts the contents we want from each link and removes any unnecessary variables on each site. Once the scraping process is finished, we prep the dataset to structure it for the correct format for training in BERT. In our case, we took a page that should be a text file and labeled it either 0 for Petro wiki or 1 for AAPG wiki. Once the dataset was formatted correctly for BERT, we trained the model to fine-tune it by adjusting the parameters as needed based on the performance of the validation set. Then, once we split the data into training and testing data, we evaluate BERT's performance metrics.

RESULTS

```python
# Loop through each file in the directory
for entry in os.scandir(directory_path):
    if entry.is_file() and entry.name.endswith('.txt'):
        file_paths[entry.name.split('.')[0]] = entry.path

# Load and preprocess the data
texts, labels = [], []
for file_name, file_path in file_paths.items():
    with open(file_path, 'r', encoding='utf-8') as file:
        label = int(file.readline().strip().split(":")[1].strip())  # Assumes label is in the first line
        text = file.read().strip()
        texts.append(text)
        labels.append(label)

data = Dataset.from_dict({"text": texts, "label": labels})

# Define the tokenizer
tokenizer = AutoTokenizer.from_pretrained('distilbert-base-uncased')

# Tokenize the dataset
def tokenize_function(examples):
    return tokenizer(examples['text'], truncation=True, padding="max_length", max_length=512)

encoded_data = data.map(tokenize_function, batched=True)

# Split the dataset into training and validation sets
train_test_split = encoded_data.train_test_split(test_size=0.1)
dataset = DatasetDict({
    'train': train_test_split['train'],
    'validation': train_test_split['test']
})
```

This code is for grabbing the data from our directory and then setting up the tokenizer and tokenizing our data set, then splitting the data into training and testing data.

The second image of code sets up our model into finding the two labels, 0 and 1. Then, we explain our arguments for the training session of the model, setup the trainer itself, and train the model.

```
Accuracy: 0.75
Classification Report:
              precision    recall  f1-score   support

    Petrowiki       0.73      0.89      0.80         9
         AAPG       0.80      0.57      0.67         7

     accuracy                           0.75        16
    macro avg       0.76      0.73      0.73        16
 weighted avg       0.76      0.75      0.74        16
```

With the accuracy score being 0.75 and the f1 score being 0.8 for Petrowiki and 0.67 for AAPG, we can say that our method performed relatively well in classifying articles from Petrowiki and AAPG. The decrease of epochs from 0.7527 to 0.6337 suggests that the model is learning from the training data.

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.752700	0.678861	0.600000	0.750000	0.600000	1.000000
2	0.662300	0.680175	0.400000	0.000000	0.000000	0.000000
3	0.633700	0.665702	0.600000	0.500000	1.000000	0.333333

```python
# Setup the model and trainer
model = AutoModelForSequenceClassification.from_pretrained('distilbert-base-uncased', num_labels=2)


training_args = TrainingArguments(
    output_dir='./results',
    evaluation_strategy='epoch',  # Evaluate at the end of each epoch
    per_device_train_batch_size=2,  # Adjust based on your GPU/CPU
    num_train_epochs=3,
    logging_dir='./logs',
    logging_strategy='epoch',  # Log at the end of each epoch
    save_strategy='epoch',  # Save at the end of each epoch
    load_best_model_at_end=True,
    metric_for_best_model='f1',
)


trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset['train'],
    eval_dataset=dataset['validation'],
    compute_metrics=compute_metrics
)

# Train the model
trainer.train()

# Evaluate the model
evaluation_results = trainer.evaluate()
print(evaluation_results)
```

The validation loss fluctuated but roughly the same from 0.66 to 0.68 saying the model is not overfitting. The high loss values suggesting the model's accuracy is limited by the smaller dataset.

The high recall 1.00 and the low precision 0.6000 reveals that the model is correctly identifying the true positives but at the cost of incorrectly labeling the false positives.

DISCUSSION

Our research faced significant challenges related to the constraints of time and resources. The limited timeframe imposed constraints on the extent to which we could expand the scope of our project and explore alternative methodologies. Consequently, we opted for a binary classification approach. A substantial portion of our project timeline was dedicated to data collection and preprocessing tasks, particularly in preparing the dataset for training the BERT model. The process of data scraping posed challenges in terms of efficiency and scale. As a result, we encountered limitations in the volume of data collected, which consequently impacted the size of our sample for both websites.

REFERENCES

1. American Association of Petroleum Geologists. (n.d.). Hydrofractured seal leakage. AAPG Wiki. Retrieved from https://wiki.aapg.org/Hydrofractured_seal_leakage

2. *Main page*. AAPG Wiki. (n.d.). https://wiki.aapg.org/Main_Page

3. PetroWiki. (2024, February 19). *An E&P industry wiki – oil&gas*. https://petrowiki.spe.org/PetroWiki

4. Saumyab271. (2024, February 13). *Text classification using Bert and tensorflow*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/12/text-classification-using-bert-and-tensorflow/

5. Society of Petroleum Engineers. (n.d.). Electromagnetic heating of oil. PetroWiki. Retrieved from https://petrowiki.spe.org/Electromagnetic_heating_of_oil

6. Society of Petroleum Engineers. (n.d.). Fluid flow in naturally fractured reservoirs. PetroWiki. Retrieved from https://petrowiki.spe.org/Fluid_flow_in_naturally_fractured_reservoirs

7. Text classification: What it is & how to get started. RSS. (n.d.). https://levity.ai/blog/text-classification#:~:text=Why%20text%20classification%20is%20important,%2C%20databases%2C%20and%20other%20documents.