CS260 SEMINAR IN COMPUTER SCIENCE: NLP Final Report

Mahitha Sannala (msann002, 862394693)
Sritha Hadassah Duddu (sdudd001, 862393175)

March 24, 2023

BERT-Based Patent Analysis

1 Abstract

This project explores the use of BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art deep learning model, for patent analysis. We apply BERT as a multi-tasking model for 3 tasks. Our first task, Document classification involves categorizing patent documents into predefined classes based on their content. We use BERT for our second task Patent Valuation which takes in abstract text attributes and gives relevance scores on market and technology keywords. We also use BERT for text similarity analysis, our third task which involves comparing the semantic similarity between patent documents. Our results show that BERT can be an effective multi-task model for patent analysis, achieving high levels of accuracy and efficiency for document classification and text similarity analysis. Until now, we have come across only single-task BERT model for patent analysis which makes our multi-task BERT model a valuable asset for patent analysis. This work has important implications for the field of patent analysis, offering new approaches for automating the analysis and organization of large volumes of patent data.

2 Introduction and Motivation

Patent analysis is critical for patent strategy, competitive intelligence, and investment decisions. NLP techniques can automate the analysis of large patent collections and extract valuable information. BERT is a powerful NLP model for patent analysis tasks such as document classification, entity recognition, and text similarity analysis. This project aims to address the challenges and limitations of traditional patent analysis methods by providing a more efficient and effective solution. Specific tasks include classifying patent documents into technology categories, identifying key entities, and detecting similarities and patterns in patent claims and specifications. The project will evaluate the performance of the BERT-based patent analysis pipeline against other NLP models through NLP evaluation metrics like accuracy, training vs validation loss curve etc...

3 Modification to the problem statement

- Our proposed tasks were Document Classification, Entity Recognition, and Text Similarity Analysis.
- We have backtracked from the Named Entity Recognition task and replaced it with Patent Valuation task. The main reason behind the modification is that we have observed that a lot of important insights can be achieved from the Patent valuation task over the NER task.
- Another important reason for the change is the lack of named entity labels to train and fine-tune the task on BERT. We first considered using spaCy library to generate labels using the "en_core_web_sm" package to generate entities but we were not satisfied with the results. For example, it generates "Photoemission" to be a "person" Entity.
- On further reference, we have come across annotating the labels manually or rule-based labelling and then Fine tune BERT model but we felt these were not efficient methodologies as the patents can contain information in vast and diverse domains.
- It was a great learning experience by trying on different approaches for the task but we have finally concluded to replace this task with a different and useful task on patents.

4 Dataset

- We initially planned to use European Patent Office (EPO) Open Patent Services (OPS) dataset but we couldn't get the developer access to extract the data. So, we had to change the dataset.
- We switched to Google Patents Public Data dataset which is free and available in google cloud and can be accessed through BigQuery.
- The dataset contains over 110 million US patents with 37 attributes and it covers all patents granted from 1976 to the present day.
- The dataset is organized into tables that include information on patent titles, inventors, assignees, patent abstracts, and other metadata. It also includes data on citations, which can be used to analyze the relationships between patents and identify emerging technologies.
- We have extracted 2 lakh rows, about 4 Gb of data for our project.
- Click here for the full dataset
- Click here for the sample dataset

5 Existing Work

- Patent Classification by Fine-Tuning BERT Language Model by Jieh-Sheng Lee and Jieh Hsiang (2019): This paper focuses on fine-tuning a pre-trained BERT model and applying it to patent classification. Patent classification is a challenging multi-label classification task with many possible labels. When applied to large datasets of over two million patents, their approach outperforms the state-of-the-art approach using CNN with word embeddings. They used USPTO-3M dataset (with 3,050,615 patents). In addition, they focus on patent claims without other parts in patent documents. Their contributions include:

  - a new state-of-the-art result based on pre-trained BERT model and fine-tuning for patent classification
  - a large dataset USPTO-3M at the CPC subclass level with SQL statements that can be used by future researchers
  - showing that patent claims alone are sufficient for classification task, in contrast to conventional wisdom
  - PatentBERT achieved F1 scores ranging from 46.85% to 66.83% depending on the dataset and label level, outperforming DeepPatent in all cases.
  - PatentBERT achieved precision ranging from 32.19% to 84.26%, and recall ranging from 53.36% to 86.06%, depending on the dataset and label level

- Named Entity Recognition for Chinese biomedical patents by Yuting Hu, Suzan Verberne (2020): The paper describes a study on Named Entity Recognition (NER) for Chinese biomedical patent data. While there have been many studies on Bio-NER for English, few studies have been done on Chinese biomedical text, and none have focused on patents.
  - the authors used pre-trained BERT models for their biomedical NER task and designed three different learning methods to train their models. The first method, called "Supervised original," fine-tuned all weights of the BERT model layers and the NER layer using a relatively small learning rate and their labeled dataset. The second method, called "LM mixed fine-tuning," first tuned the weights of the BERT language model layers with their unlabeled dataset and then repeated the supervised original learning step. The third method, called "PartBERT+CRF fine-tuning," fine-tuned the weights of part of the BERT model (last 4 layers) plus an added CRF layer, trained with their labeled dataset.
  - the authors contribute by creating a labeled dataset of 5,813 sentences and 2,267 named entities from 21 Chinese biomedical patents and evaluating the performance of a BERT-based NER model on this dataset.
  - To make the training process more efficient, the authors created two smaller subsets out of their two large unlabeled datasets to fine-tune the BERT model: "partBC" and "partHG." Both subsets contained a train and test set to train the language models.
  - the optimized model (BERT LM mixed (partHG 1epoch)) achieved an F1 score of 0.54 ± 0.15, indicating promising results for identifying genes, proteins, and diseases in Chinese biomedical patents.

- "PatentBERT: Patent Classification using BERT" by Xu et al. (2021): This paper proposed a patent classification model based on BERT and evaluated its performance on the WIPO-alpha dataset. They fine-tuned the BERT model on a large set of patent documents and used a hierarchical classification approach to categorize patents into their respective technology classes. They demonstrated that their approach outperformed existing state-of-the-art methods.

- Overall, these studies demonstrate the potential of using BERT for various tasks related to patent analysis, including classification, clustering, and similarity analysis.

6 Pre-Processing

- As a part of pre-processing for our task, we have first extracted the titles, abstract, and IPC codes. The IPC codes contain 8 categories, the dictionaries {A: Human Necessities, B:

Performing Operations; Transporting, C: Chemistry; Metallurgy; D: Textiles; Paper; E: Fixed Constructions; F: Mechanical Engineering; Lighting; Heating; Weapons; Blasting; G: Physics; H: Electricity}.

- The IPC codes from the first 25000 patents have been extracted, the first letter of IPC codes "A", "B", "C","D" ,"E", "F", "G", "H" helps us to classify the document into "different categories" with the help of the pre-processed title text.
- As a part of pre-processing the title and abstract text, we first extracted the english text from the data frame using the json loads and then removed or cleaned stopwords and specific characters, lowercased and stemmed the text.
- As a part of pre-processing the IPC codes, after extracting the first letter, there can be multiple IPC codes for a single patent as it can belong to a different domain, we have created a list to store these labels, for example, if a patent belongs to category A and C, it label encoded list will be [1,0,1,0,0,0,0,0].

7 Task-1: Document Classification

7.1 Introduction

- Document classification is the process of automatically categorizing a document into one or more predefined categories based on its content. The goal of document classification is to automate the task of sorting large volumes of text documents into different categories, which can be useful for various applications such as information retrieval, topic modeling, sentiment analysis, and spam filtering.
- BERT (Bidirectional Encoder Representations from Transformers) is a powerful language model that has been successfully used for a variety of natural language processing (NLP) tasks, including document classification. In the context of patent analysis, BERT can be used to classify patent documents into different categories based on their content.
- The basic idea behind document classification with BERT is to train a model to predict the category of a given document based on its text. Since classification is part of BERT, this doesn't involve fine-tuning the BERT model.
- After the model is trained, it can be used to classify new patent documents by feeding their preprocessed text into the model and predicting their category based on the output of the model.

7.2 Uses

After running the model (input "title text"), it will classify them into relevant categories namely A: Human Necessities, B: Performing Operations; Transporting, C: Chemistry; Metallurgy; D: Textiles; Paper; E: Fixed Constructions; F: Mechanical Engineering; Lighting; Heating; Weapons; Blasting; G: Physics; H: Electricity. Document classification can be used to improve the accuracy and efficiency of information retrieval systems, by categorizing documents based on their content and making it easier to search and find relevant documents.

7.3 Explanation of code

- We ran on 1000 documents due to the computation availability on Collab. As of now, we have label encoding column as a target variable, and the title text as an input, We have used BertTokenizer and pre-trained BERTforSequenceClassification model which outputs 8 labels.
- As a part of model training, we have first split the model into train and validation and for testing, we have considered the next 100 documents from 1001 to 1100.
- We first tokenized the input data into a training and validation set and converted the input data to PyTorch tensors for them. TensorDataset is created for the training and validation set. A batch size of 8 is considered and a dataloader is used to load the data in batches.
- Adam Optimizer and binary cross-entropy loss function are used for multi-label classification for 4 epochs. The model is trained on the training set and evaluated on the evaluation set.

7.4 Code snippet

```
# Tokenize input data for training set
train_tokenized_texts = [tokenizer(text, padding=True, truncation=True, max_length=128) for text in train_texts]
# Tokenize input data for validation set
val_tokenized_texts = [tokenizer(text, padding=True, truncation=True, max_length=128) for text in val_texts]

# input data to PyTorch tensors for training set
train_input_ids = pad_sequence([torch.tensor(tokenized_text['input_ids']) for tokenized_text in train_tokenized_texts],
                               batch_first=True, padding_value=0)
train_attention_masks = pad_sequence([torch.tensor(tokenized_text['attention_mask']) for tokenized_text in train_tokenized_texts],
                                     batch_first=True, padding_value=0)
train_labels = torch.tensor(train_labels)

#  input data to PyTorch tensors for validation set
val_input_ids = pad_sequence([torch.tensor(tokenized_text['input_ids']) for tokenized_text in val_tokenized_texts],
                               batch_first=True, padding_value=0)
val_attention_masks = pad_sequence([torch.tensor(tokenized_text['attention_mask']) for tokenized_text in val_tokenized_texts],
                               batch_first=True, padding_value=0)
val_labels = torch.tensor(val_labels)

# TensorDataset for the training set
train_dataset = TensorDataset(train_input_ids, train_attention_masks, train_labels)

# TensorDataset for the validation set
val_dataset = TensorDataset(val_input_ids, val_attention_masks, val_labels)

# DataLoader to load the training set in batches
batch_size = 8
train_dataloader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)

# a DataLoader to load the validation set in batches
val_dataloader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)

# Train model
optimizer = torch.optim.Adam(model.parameters(), lr=1e-5)
loss_fn = torch.nn.BCEWithLogitsLoss() # binary cross-entropy loss for multi-label classification
num_epochs = 4

for epoch in range(num_epochs):
    model.train() #train the model
    epoch_loss = 0
    for batch in train_dataloader:
        batch_input_ids, batch_attention_masks, batch_labels = batch
        optimizer.zero_grad()
        outputs = model(batch_input_ids, attention_mask=batch_attention_masks, labels=batch_labels.float())
        loss = loss_fn(outputs.logits, batch_labels.float())
        loss.backward()
        optimizer.step()
        epoch_loss += loss.item()
    print(f"Epoch {epoch + 1} train loss: {epoch_loss / len(train_dataloader)}")

    # Evaluate model on validation set
    model.eval()
    val_loss = 0
    with torch.no_grad():
        for batch in val_dataloader:
            batch_input_ids, batch_attention_masks, batch_labels = batch
            outputs = model(batch_input_ids, attention_mask=batch_attention_masks, labels=batch_labels.float())
            loss = loss_fn(outputs.logits, batch_labels.float())
            val_loss += loss.item()
    print(f"Epoch {epoch + 1} val loss: {val_loss / len(val_dataloader)}")
```

Figure 1: code snippet

7.5 Output

To calculate the results on test data, we have extracted 100 rows of data from 1000 to 1100, the input is the text data of the titles, and this input data is converted to PyTorch tensors for test data and passed into the model which is predicted using the sigmoid function as it gives probability values between 0 to 1 to denote which class it is more probable too.

```
new_texts = list(X_test)
new_tokenized_texts = new_tokenized_texts = [tokenizer(text, add_special_tokens=True, max_length=16,
                                      padding='max_length', truncation=True) for text in new_texts]
new_input_ids = pad_sequence([torch.tensor(tokenized_text['input_ids']) for tokenized_text in new_tokenized_texts], batch_first=True, padding_value=tokenizer.pad_token_id)
new_attention_masks = pad_sequence([torch.tensor(tokenized_text['attention_mask']) for tokenized_text in new_tokenized_texts], batch_first=True, padding_value=0)
new_outputs = model(new_input_ids, attention_mask=new_attention_masks)
new_predictions = torch.sigmoid(new_outputs.logits).tolist()

print(new_predictions)
```

```
[[0.5052585005760193, 0.14596988260746002, 0.6258080005645752, 0.06700287759304047, 0.07629520446062088, 0.10476664453744888, 0.194210484623909, 0.11589989066123962], [0.21566762030124664, 0.29152524471
```

```
y_pred =[]
for i in range(len(new_predictions)):
  l=[0]*8
  maxi = max(new_predictions[i])
  ind = new_predictions[i].index(maxi)
  for j in range(8):
    if maxi <= 0.5 :
      l[ind] = 1
    else:
      if new_predictions[i][j] >= 0.5:
        l[j] = 1
  y_pred.append(l)
```

Figure 2: code for output prediction and converting them to 0 and 1s

7.6 Results and graphs metrics evaluation

```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, y_pred, average='micro')
print(f1)
```

```
0.45614035087719296
```

Figure 3: f1 score for the document classification task

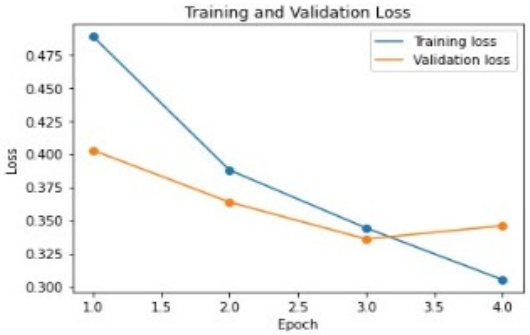

Figure 4: Training vs validation log loss

8 Task-2: Patent Valuation

8.1 Introduction

- Patent valuation is the process of assessing the monetary value of a patent or portfolio of patents. It involves estimating the economic benefits that can be derived from the patent, taking into account factors such as the market size, potential licensing revenue, and the patent's scope of protection.
- In the context of patent valuation, cosine similarity is used to compare the text content of different patents and determine their similarity, which can be a useful tool for patent valuation.

8.2 Uses

Patent valuation can be used to assess the value of a company's patent portfolio and help identify patents that are no longer valuable or relevant. This can help companies optimize their patent portfolio and focus on the most valuable patents.

8.3 Explanation of code

- In this task, we have considered the first 1000 documents to score them based on market and technical keywords. We first made a list for market which contains words like product, user, consumer, commercial, sector, service,vand their synonyms, and another list with technical keywords like innovation, creative, engineering, design, and their synonyms.
- Now, our primary task is to create relevant scores for our patents so as to know their valuation in the market.
- First defined a function to encode the text using the BERT tokenizer and model and then a function to calculate the relevance score of a text based on the cosine similarity between the embeddings of the text and the embeddings of the relevant keywords and defined an extract_info function to find the relevant words from the abstract in order to calculate the scores.
- The higher the score of relevance for market and technology, we can conclude the higher the valuation.

8.4 Code snippet

```
# function to encode the text using the BERT tokenizer and model
def encode_text(text):
    input_ids = torch.tensor(tokenizer.encode(text, add_special_tokens=True)).unsqueeze(0)
    with torch.no_grad():
        outputs = model(input_ids)
        embeddings = outputs[0][:, 0, :].numpy()
    return embeddings

# function to calculate the relevance score of a text based on the cosine similarity between the embeddings of the text and the embeddings of the relevant keywords
def calc_score(text, keywords):
    text_embedding = encode_text(text)
    keyword_embeddings = [encode_text(keyword) for keyword in keywords]
    scores = [np.dot(text_embedding[0], keyword_embedding[0])/(np.linalg.norm(text_embedding[0])*np.linalg.norm(keyword_embedding[0])) for keyword_embedding in keyword_embeddings]
    return sum(scores)

# function to extract the relevant information from the abstract of a patent document
def extract_info(patent_doc):
    abstract = patent_doc
    market_score = calc_score(abstract , market_keywords)
    tech_score = calc_score(abstract , tech_keywords)
    return market_score, tech_score

market_score_lis = []
tech_score_lis = []

for i in range(len(new_df)):
  print(i)
  m_score, t_score = extract_info(list(new_df ["abstract text"])[i].lower())
  market_score_lis.append(m_score)
  tech_score_lis.append(t_score)
```

Figure 5: code for calculating the scores for market and tech valuation

8.5 Results

title text	prepocess title text	abstract text	prepocess abstract text	codes from cpc	codes from ipc	labels from ipc	labels encoding	market scores	technology scores
Replacement cardiac valves and methods of use ...	replac cardiac valv method use manufactur	Prosthetic mitral valves and their methods of ...	prosthet mitral valv method manufactur use	[A61F2250/0018, A61F2220/0041, A61F2/2409, A61...	[A61F2/24]	[Human Necessities (including Medical)]	[1, 0, 0, 0, 0, 0, 0]	13.229822	9.105388
Photoemission monitoring of EUV mirror and mas...	photoemiss monitor euv mirror mask surfac cont...	Photoelectron emission mapping systems for use...	photoelectron emiss map system use euv extrem ...	[G01N2223/611, H01L31/02168, G01N2223/652, G03...	[G01N21/94, G01N21/956, G01N23/227, H01L31/021...	[Physics, Electricity]	[0, 0, 0, 0, 0, 1, 1]	7.477611	5.202711
Fibrous structures	fibrou structur	A fibrous structure includes first, second, an...	fibrou structur includ first second third rel ...	[D21F11/14, D21F11/006, Y10T428/24628, D21H27/...	[D21F7/08, D21H27/02, D21F11/00, D21F11/14, D2...	[Textiles, Paper]	[0, 0, 0, 1, 0, 0, 0, 0]	12.288726	8.229364
Airborne component extractor with adjustable f...	airborn compon extractor adjust flow rate	A extraction system is designed for metal work...	extract system design metal work applic system...	[B08B15/00, F24F7/007, B08B15/02, B23K26/142, ...	[F24F7/007, F24F11/00, F24F7/00, B23K37/08, B0...	[Mechanical Engineering, Lighting, Heating, We...	[0, 1, 0, 0, 0, 1, 0, 0]	10.089169	6.901220
Data loss prevention (DLP) methods by a cloud ...	data loss prevent dlp method cloud servic incl	Embodiments of the present disclosure include ...	embodi present disclosur includ data loss prev...	[G06F21/6227, G06F40/197, G06F21/554, G06F16/1...	[G06F21/60, G06F17/00, G06Q10/10, H04L29/06, G...	[Physics, Electricity]	[0, 0, 0, 0, 0, 0, 1, 1]	10.247713	6.974740

Figure 6: The output tech and market scores based on abstract on Dataframe

9 Task-3: Similarity Analysis

9.1 Introduction

- Similarity analysis refers to the process of quantifying the degree of similarity or relatedness between two or more pieces of text data. The goal of similarity analysis is to identify the extent to which two pieces of text share common themes, topics, or ideas and can be useful for a variety of applications such as text clustering, document retrieval, and plagiarism detection.
- Text similarity analysis using BERT for patent classification involves using pre-trained BERT-based models such as Roberta (a variant of BERT) in conjunction with Faiss (a similarity search library) to measure the similarity between patent documents through similarity score.
- The patent documents are first encoded as vectors using Roberta. The vectors are then indexed using Faiss to enable a fast similarity search. When a new document is submitted for similarity analysis, it is also encoded as a vector using the same Roberta model and then compared to the existing document vectors using Faiss to retrieve the most similar documents.
- We have considered only a part of the dataset(25000 patents) and the patent was queried in the index gives the top k results based on the similarity score.
- 1st result is removed as it would be the query patent itself and the remaining results are the patents similar to the given patent.

9.2 Uses

There are several potential uses of text similarity analysis using BERT for patent classification with Roberta and Faiss, for example, Companies can use text similarity analysis to manage their patent portfolios by identifying patents that are similar to each other. This can help identify potential areas of overlap or redundancy in the portfolio and inform strategic decision-making such as patent licensing or acquisition.

9.3 Explanation of code

9.3.1 Indexing using index_generator.ipynb

- Preparing the Data: The data was cleaned and turned into a BERT-compatible format. This step is critical because it guarantees that the input data is in a standardized format that the model can use.
- Indexing: The next step was to generate an index of the patent abstract attributes. An index is a type of data structure that enables the quick and easy retrieval of documents based on specified keywords or phrases. The patent abstract attributes were indexed using BERT embeddings.
- bert-patent-abstract.index is the index file generated.

9.3.2 bertquery.py file

- To use the BERT index for ranking, load the index created in the previous step.

- When a user types a query, we use BERT to encode it into a vector representation. We next use similarity metric, such as Euclidean distance, to compare this query embedding to the embeddings of the indexed patents.
- Each patent receives a similarity score, showing how well it fits the query. Lastly, we rank the patents based on their similarity ratings, with the ones that are the most similar being ranked first.
- In brief, utilizing the BERT index for ranking entails using BERT to encode both the query and the patent abstract attributes into vector representations, computing similarity scores, then rating them based on those scores.

9.4 Code snippet

```
def convert_to_embedding(query):
    query = removeStopWords(query)
    tokens = {'input_ids': [], 'attention_mask': []}
    new_tokens = tokenizer.encode_plus(query, max_length=512,
                                       truncation=True, padding='max_length',
                                       return_tensors='pt')
    tokens['input_ids'].append(new_tokens['input_ids'][0])
    tokens['attention_mask'].append(new_tokens['attention_mask'][0])
    tokens['input_ids'] = torch.stack(tokens['input_ids'])
    tokens['attention_mask'] = torch.stack(tokens['attention_mask'])
    with torch.no_grad():
        outputs = model(**tokens)
    embeddings = outputs.last_hidden_state
    attention_mask = tokens['attention_mask']
    mask = attention_mask.unsqueeze(-1).expand(embeddings.size()).float()
    masked_embeddings = embeddings * mask
    summed = torch.sum(masked_embeddings, 1)
    summed_mask = torch.clamp(mask.sum(1), min=1e-9)
    mean_pooled = summed / summed_mask

    return mean_pooled[0] # assuming query is a single sentence
```

Figure 7: code for converting to embeddings

```
def ranking(index, count):
    index_loaded = faiss.read_index('bert-patent-abstract.index')

    sentences = pd.read_csv("patent_abstract.csv", header=None)
    data = sentences[0].astype(str).tolist()
    data.pop(0)
    load = pd.read_csv("patent_title.csv", header=None)
    patents = load.iloc[1:]
    patents.index = np.arange(0, len(patents))

    patent = patents.loc[index]
    print("Query:", patent[0])
    query = data[index]
    query_embedding = convert_to_embedding(query)
    res = index_loaded.search(query_embedding[None, :], count)
    D, I = res
    indexes = list(I[0])
    score = list(D[0])
    indexes.pop(0)
    score.pop(0)
    ranks = pd.DataFrame()
    ranks['patents'] = patents.loc[indexes]
    ranks['scores'] = score
    return ranks
```

Figure 8: code for ranking the documents

9.4.1 Instructions to run

To run bertquery.py:

python3 bertquery.py -q "patent-index-from-csv-file" -c "number-of-required-results(k)"

This fetches the top k similar patents to the query along with their similarity scores. We have already created an index file "bert-patent-abstract.index" for a part of the dataset(25000 patents), hence you would just have to run bertquery.py making sure that these below files are in the same folder:

- patent_abstract.csv
- bertquery.py
- bert-patent-abstract.index
- patent_title.csv

9.5 Results

These are the examples that we ran. In the first example, we set the number of required results to 5, so it gives the top 5 similar patents based on the similarity score. In the second example, we tried a different patent and set the number of required results to 10, so it gives the top 10 similar patents to the query.

```
Starting Patent Similarity Query:-

Query: replacement cardiac valves methods use manufacture

Top 5 similar patents:
                                                patents     scores
21295  antimicrobial lenses processes prepare methods...  67.921715
10459                                   antibodies tau  64.895256
11241  compounding systems methods safe medicament tr...  61.450420
22827  maxillofacial implant injection molding metal ...  60.987782
14949  valve prosthesis replacing atrioventricular va...  59.691433
```

Figure 9: Top 5 similar patents to the query 1 with their similarity scores

```
Starting Patent Similarity Query:-

Query: method device certifying compliance software resources common interface standard

Top 10 similar patents:
                                              patents     scores
18438  methods systems verification vehicle controlli...  38.426098
22435           infectious bursal disease virus variant  32.722141
22635  method apparatus providing active compliance p...  32.532394
15089  limiting transactions based lack proximity ass...  30.876774
18688                          methods defect validation  30.548420
24820  partial discharge detection device capable det...  29.674347
22558  method detecting disconnection extracorporeal ...  29.486191
18655               method producing reagent instrument  29.434010
18346  plateable thick film silver paste plating proc...  28.570299
14734  method apparatus collecting electronic signatu...  28.279278
```

Figure 10: Top 10 similar patents to the query 2 with their similarity scores

10 Challenges faced

- Processing the data to an index using BERT takes a long time and hence we were able to do it only for part of the dataset.
- Patent documents can contain sensitive information, such as trade secrets or confidential business information. As a result, there may be legal or ethical concerns around using BERT for patent analysis, particularly if the analysis involves sharing or disclosing patent data.
- Patent documents may contain noise, errors, or inconsistencies, which can make it challenging to preprocess the text data in a way that is suitable for BERT analysis.

11 Conclusion

We performed three tasks for Patent Analysis on BERT, namely Document classification, patent valuation and patent similarity scores and were able to achieve good and relevant results on the three tasks. Though, increasing the dataset and further fine tuning can improve the results but the current results on a small cluster due to our computational restrictions have shown reliable results.

12 Future Work

- As part of improvisation to the above code for document classification, we have come across a more detailed categorization column called newly added CPC codes which is similar to IPC codes but has much more detailed sub-categorization which can be helpful to make many elaborated categories.
- As a part of improvisation to the above code for Patent valuation, we have come across BERT for regression (Camembert Model), where we can fine-tune the model to learn scores based on

the abstract text and make predictions of relevant scores based on input abstract text.

- As a part of improvisation to the above code for similarity analysis, We can set a certain threshold, and using this threshold; we can determine if the returned documents are actually similar or not. If the score is greater than the threshold, then the patent is similar else it is unique enough.
- As a part of improvisation to the above code for similarity analysis, If we set the threshold at 50, we can see that there are a number of patents similar to the 1st patent, but for the last patent, there are no similar patents because the scores are all less than 50.