OPTIMIZING PRIOR ART SEARCH AND EPO SEARCH REPORT GENERATION USING LLM

EPO Codefest 2024

Team Name: Confused Electrons

Team members:

- Mauricio Rodriguez Alas
- Arnab Saha
- Ralph Ryan Hebrio
- Trần Lê Phương Lan

OCTOBER 23, 2024

Chapter 1: Introduction

Patents are the foundation of innovation, protecting inventors and encouraging technological progress. However, the journey from idea to granted patent is complex and rigorous, involving numerous assessments and evaluations.

To support this journey, the European Patent Office (EPO) search reports are crucial in providing applicants with an assessment of prior art relevant to their invention. They provide applicants with an evaluation of prior art relevant to their invention. Traditionally, generating these reports involves extensive manual effort, including searching through large patent databases and classifying information according to the International Patent Classification (IPC) system. This process can be time-consuming, costly, and prone to inconsistencies. Below is an overview of the Patent Application process:

1.1 Patent Application Process Overview:

The patent application process can be broken down into the following steps:

- **Submission of Application:** The process begins with the submission of a patent application, which includes a detailed description of the invention, claims, and any necessary drawings.

- **Formal Examination**: After submission, the application undergoes a formal examination to ensure that all required documents are included and comply with procedural rules. This step ensures that the application is complete and adheres to the necessary formalities.

- **Search Report Generation:** Once the formal examination is complete, a search report is generated to identify relevant prior art that may affect the patentability of the invention. The EPO search report is essential for assessing whether the invention is novel and involves an inventive step.

- **Substantive Examination:** The application proceeds to a substantive examination, during which the invention is evaluated for novelty, inventive step, and industrial applicability. The search report plays a key role in this evaluation.

- **Initial Opinion and Communication:** The patent examiner issues an initial opinion based on the search report and examination results. This may lead to further communications with the applicant, during which the applicant can amend the claims or provide arguments in response to any objections raised by the examiner.

- **Grant or Refusal:** If all objections are addressed and the examiner is satisfied that the invention meets the patentability requirements, the patent is granted. If the requirements are not met, the application may be refused.

This process can be lengthy and involves multiple stages of review and communication between the applicant and the patent office. After the submission of the application, the applicants need to wait for the EPO search report to know about the state of their application. Below is a small overview of the details the EPO search report contains:

1. 2 Details of EPO Search Report:

The EPO search report is a document that identifies prior art relevant to the claimed invention. It serves as a critical reference for assessing whether the invention meets the requirements of novelty and inventive step. The key components of an EPO search report include:

- **Search Strategy:** The report outlines the search strategy used by the examiner, including the databases and keywords utilized to locate relevant prior art. This transparency helps applicants understand how the search was conducted and provides insights into the thoroughness of the examination.

- **Citations of Prior Art:** The report lists all relevant prior art documents found during the search. These citations are categorized based on their relevance to the different claims of the patent application. The documents may include patent literature as well as non-patent literature (e.g., scientific articles).

- **Category Codes:** Each cited document is assigned a category code that indicates its relevance. Common category codes include:

    - **X:** The document is highly relevant and may challenge the novelty of the claimed invention.

    - **Y:** The document may challenge the inventive step when combined with other references.

    - **A:** The document is of general interest and may provide background information.

- **Search Report Opinion:** In some cases, the EPO search report includes a preliminary opinion on patentability. This opinion provides an initial assessment of whether the invention meets the patentability requirements, including novelty and inventive steps. This can help applicants determine the likelihood of success for their application and guide further actions.

- **Claims Analysis:** The report typically includes an analysis of how each claim is affected by the identified prior art. This analysis helps applicants understand which aspects of their invention may need modification or clarification to overcome objections.

The EPO search report plays a key role in guiding both the patent examiner and the applicant throughout the examination process. It helps ensure that the invention meets the patentability criteria and provides a foundation for any subsequent amendments or arguments by the applicant. However, the examiners face some challenges in generating the EPO search report which is overviewed below:

1.3 Challenges in Creating EPO Search Reports:

Analyzing a patent application and generating the search report places some challenges in front of the Patent Examiner as follows:

- **Volume of Prior Art:** Patent examiners must search through vast amounts of prior art, including both patent and non-patent literature. This can be overwhelming and time-consuming, especially with the growing volume of new patents and publications.

- **Complexity of Inventions:** Many patent applications involve complex technical concepts that require specialized knowledge. Understanding these inventions and determining the relevant prior art can be challenging, particularly when the examiner does not have expertise in the specific field.

- **Ambiguity in Claims:** Patent claims can often be ambiguous or overly broad, making it difficult for examiners to determine the scope of the invention and identify relevant prior art. This can lead to challenges in accurately assessing the novelty and inventive step.

- **Time Constraints:** Examiners are typically under significant time pressure to complete search reports. Balancing the need for thoroughness with the need to meet deadlines can lead to challenges in ensuring the quality and completeness of the search.

- **Language Barriers:** Prior art can be published in multiple languages, which may require translation or interpretation. This adds an additional layer of complexity to the search process, particularly when the relevant prior art is in a language unfamiliar to the examiner.

1.4 Objectives of the Project:

In this Project we tried to solve this problem by leveraging the power of Generative AI to optimize the search report generation process. Below is an overview of what we tried to do in this project:

- **Goal 1: Utilize GenAI Models** The primary goal was to use Generative AI models to create EPO search reports, leveraging advanced NLP techniques to streamline the process.

- **Goal 2: Improve Classification Accuracy** We focused on improving the accuracy of classifying relevant prior art, ensuring that the search reports are reliable and comprehensive.

- **Goal 3: Automate Prior Art Searches** By automating the search for prior art, we aimed to reduce the manual effort required by examiners, thereby increasing efficiency.

- **Goal 4: Assist Patent Examiners** Our objective was to effectively support patent examiners by simplifying the examination process and minimizing the complexity involved in prior art searches.

- **Goal 5: Shorten the Timespan for Report Generation** A key goal was significantly reducing the time required to generate an EPO search report, accelerating the overall patent application process.

Chapter 2: Methodology

2.1 Dataset Preparation

2.1.1 Data Source

The dataset used in this project consists of patent data from the European Patent Office (EPO) and other patent databases. The dataset includes fields such as publication.number, title, claims, IPC codes, etc. We extracted this data using the **EPABClient** from **`epo.tipdata.epab`**, which streamlined our querying process. However, we only used the claims and IPC for model training due to resource limitations. The `Data Preprocessing.ipynb` also includes functions to preprocess titles, descriptions, and abstracts for future optimization when more resources are available.

2.1.2 Dataset for IPC Classification

To ensure that the input data is properly formatted and prepared for model training, we implemented a comprehensive data loading and cleaning process. This process involved querying the EPAB patent database, cleaning the International Patent Classification (IPC) codes and claims, and generating a training dataset. The steps involved are as follows:

Data Loading:

- We used the **EPABClient** library to query patent data from the European Patent Office (EPO) database.

- The queried data included columns such as publication_number, IPC codes, and claims.

Data Cleaning:

- After querying the data, we performed several cleaning steps to ensure the dataset was usable for model training.

- We extracted and cleaned the IPC codes by removing unwanted characters and retaining only the relevant alphanumeric symbols and slashes.

- We filtered the claims to keep only those written in English and cleaned the text by removing HTML tags, claim numbers, and unwanted special characters.

Dataset Generation:

- Once the data was cleaned, we loaded the data in two ways: first by randomly selecting a specific amount of data, and second by ensuring even distribution across IPC sections. After training the model on both datasets, we found that the random distribution performed better due to the higher availability of data.

- We prepared it for model training by generating two datasets from it (Train Dataset- 80%, Evaluation Dataset - 20%)

- **Train Dataset:** We split the cleaned data into training and testing datasets, using 90% of the data for training and 10% for testing.

- **Evaluation Dataset:** The Evaluation dataset was kept unseen from the model during the training. This dataset was used to load the trained model afterward and check its performance.

Below is a detailed flow of the data preprocessing steps implemented in **Data Preprocessing.ipynb** to prepare the dataset for the model training and evaluation:

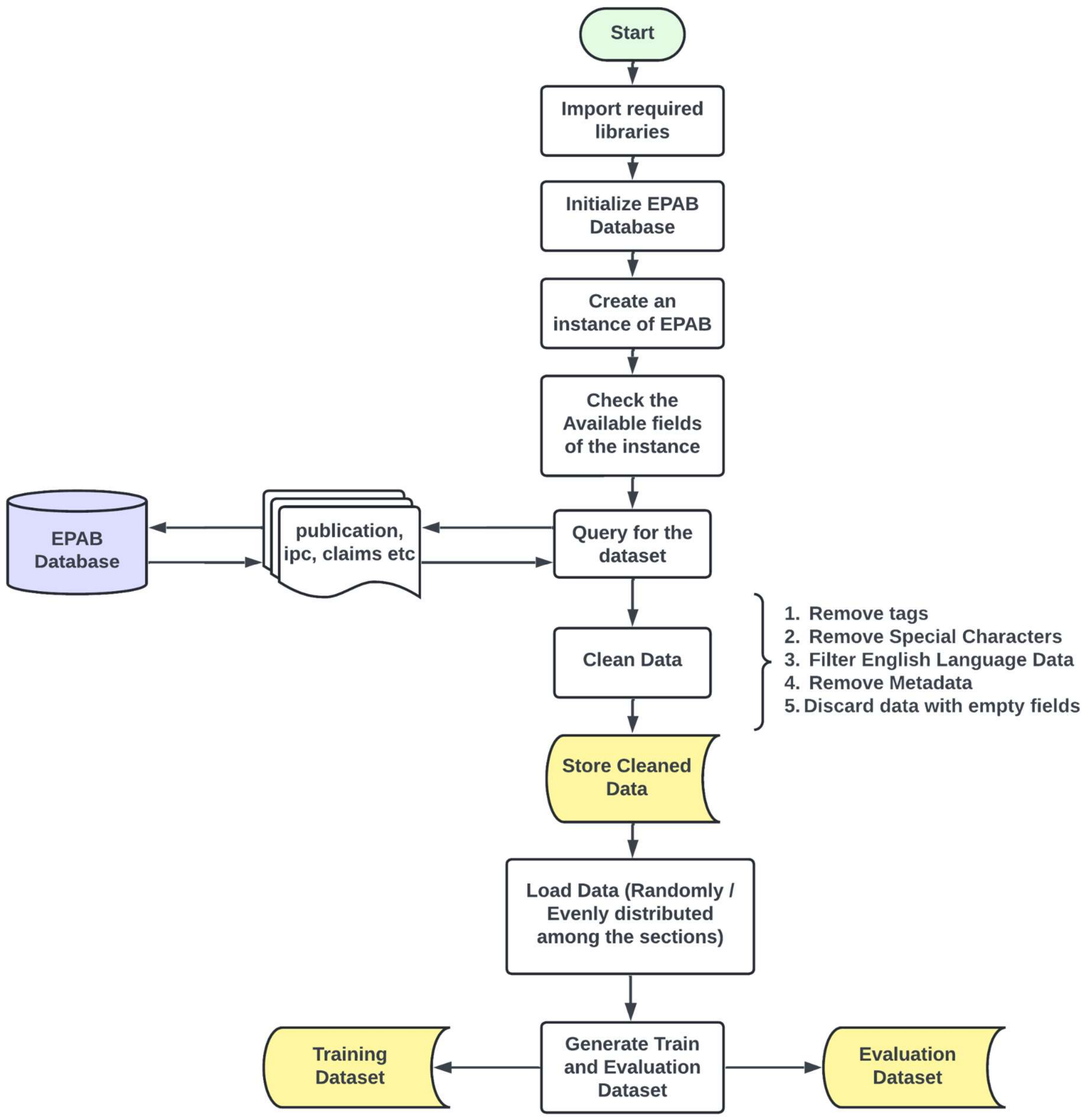

Start
Import required libraries
Initialize EPAB Database
Create an instance of EPAB
Check the Available fields of the instance
EPAB Database
publication, ipc, claims etc
Query for the dataset
Clean Data
1. Remove tags
2. Remove Special Characters
3. Filter English Language Data
4. Remove Metadata
5. Discard data with empty fields
Store Cleaned Data
Load Data (Randomly / Evenly distributed among the sections)
Training Dataset
Generate Train and Evaluation Dataset
Evaluation Dataset

Figure 2.1.1: Flow of Data Preprocessing

2.1.3 Dataset for Prior Art Generation

In addition to preparing the dataset for IPC classification, we also generated a dataset specifically for prior art searches. This dataset was designed to retrieve relevant prior art documents based on the patent application data.

Data Query:

- We queried the EPO database using **EPABClient** to retrieve the following fields: publication.number, application.filing_date, IPC codes, and details from the search report, including whether there were unity issues, incomplete searches, and relevant citations.

Data Cleaning:

- We applied similar data cleaning steps to the prior art dataset as used for the IPC classification dataset, such as cleaning the IPC codes and extracting useful citation information.

- For each document in the srep_citation field, we extracted and cleaned patent document identifiers from the search report citations.

Dataset Generation:

- The final cleaned prior art data, including the publication_number, IPC codes, search report IPC fields, and srep_citation, was saved to a CSV file.

- Additionally, we extracted document_numbers from the citations, which were saved to a separate file. We then used that csv file to query each of these documents from the EPAB to create a dataset for PriorArt embeddings. If certain documents were not part of the EPAB client, they were skipped.

2.2 Model Architecture:

Before utilizing the transformer-based PatentBERT model, we experimented with three traditional machine learning classifiers: K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression. This step provided a benchmark for understanding the classification performance on patent claims before transitioning to more advanced models.

Classifier Comparison:

- **K-Nearest Neighbors (KNN):** KNN, a non-parametric method, classifies instances based on the majority class of their nearest neighbors. However, it struggled with the high-dimensional patent claims data and required substantial memory, especially for larger datasets. We tuned the number of neighbors (k) to check the model performance. The lowest n = 3 gave the best performance.

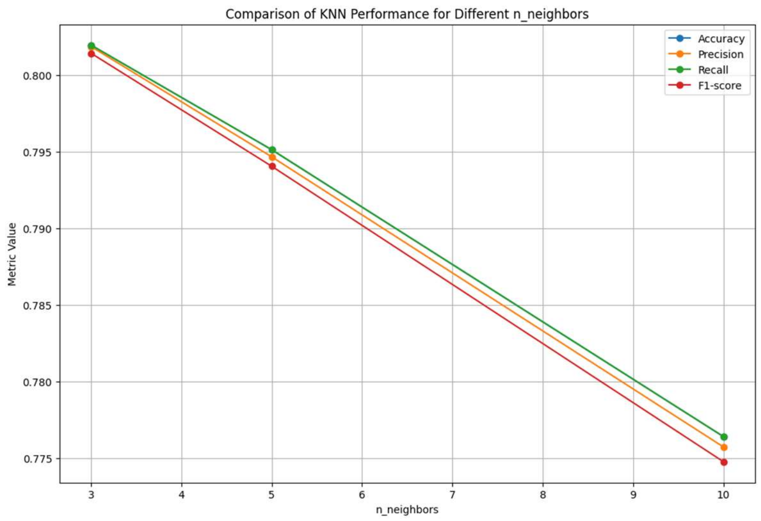


Figure: Comparison of kNN performance

Below is the confusion matrix of the best kNN model (n=3):

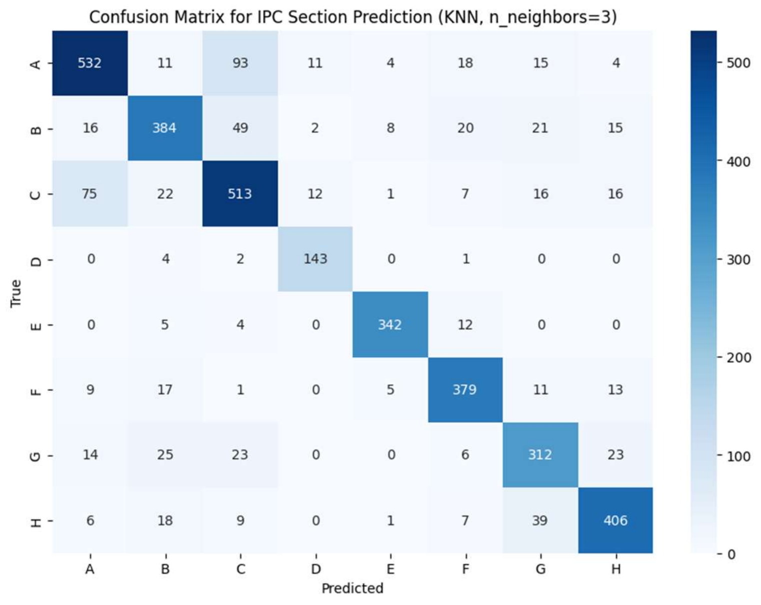


Figure: Confusion matrix for kNN (n=3)

- **Support Vector Machine (SVM):** SVM constructs hyperplanes to separate classes and is effective for high-dimensional data. It performed slightly better than Logistic Regression, but required more computational time, especially for large datasets. We tested linear, RBF, and polynomial kernels, with the polynomial (poly) kernel showing the best accuracy by capturing complex patterns in the data.

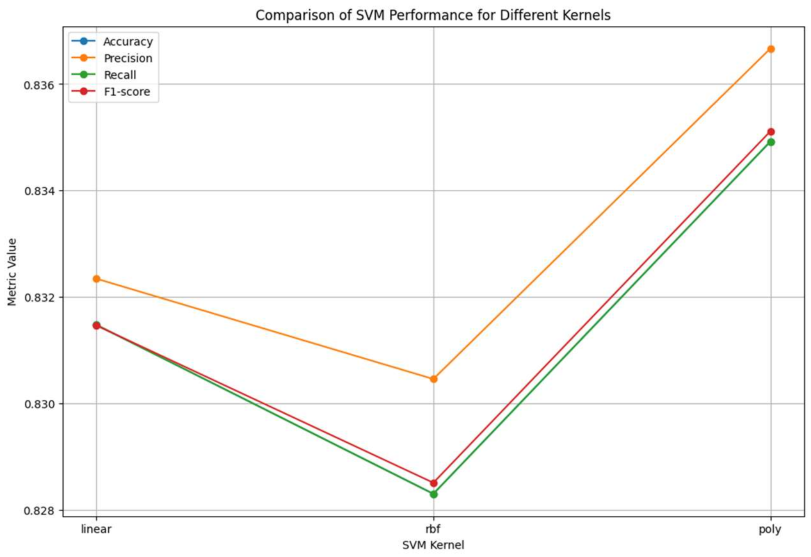
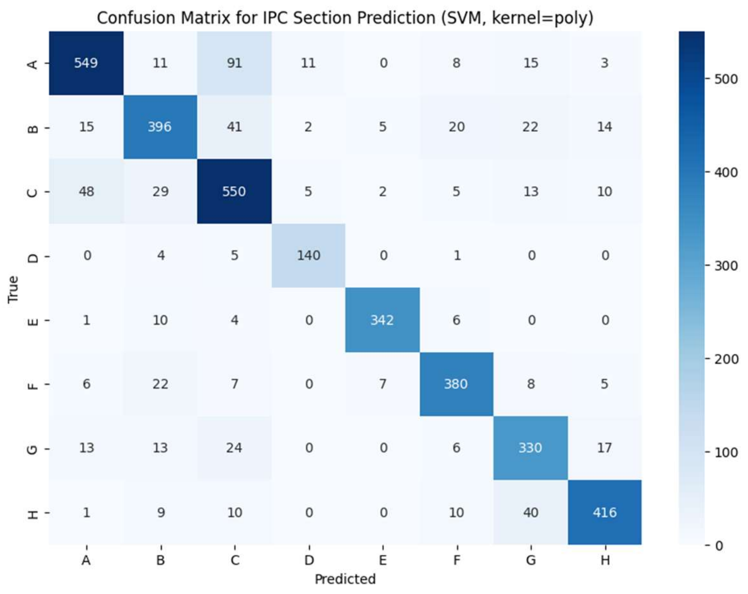


Figure: Comparison of SVM performance

Below is the confusion matrix of SVM (poly)



	A	B	C	D	E	F	G	H
A	549	11	91	11	0	8	15	3
B	15	396	41	2	5	20	22	14
C	48	29	550	5	2	5	13	10
D	0	4	5	140	0	1	0	0
E	1	10	4	0	342	6	0	0
F	6	22	7	0	7	380	8	5
G	13	13	24	0	0	6	330	17
H	1	9	10	0	0	10	40	416

Figure: Confusion matrix for SVM (poly)

- **Logistic Regression:** Logistic Regression uses a linear approach to classification, making it both computationally efficient and interpretable. It performed similarly to SVM, with only marginally lower accuracy, but it was much faster and easier to implement. Logistic Regression showed improved performance as the regularization parameter C increased. After testing various values, we finalized C=100, which provided the best balance between accuracy and overfitting, resulting in better model performance compared to lower C values.

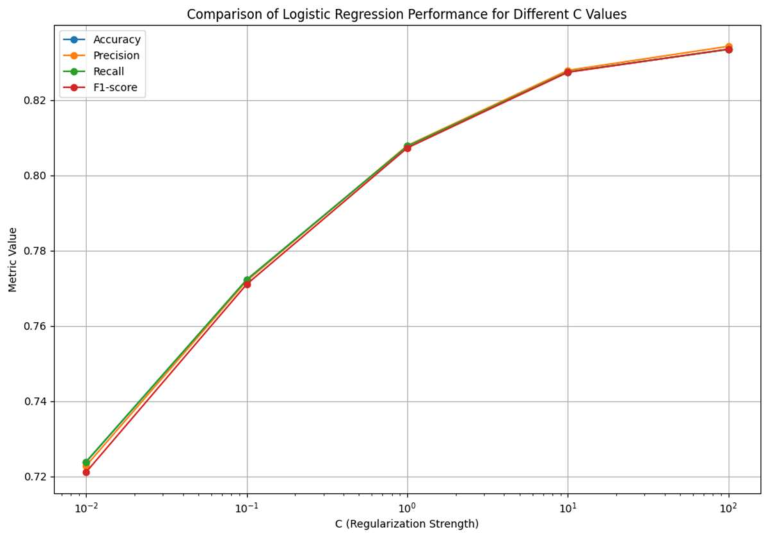


Figure: Performance of LogisticRegression

Below is the confusion matrix for Logistic regression (C=100)

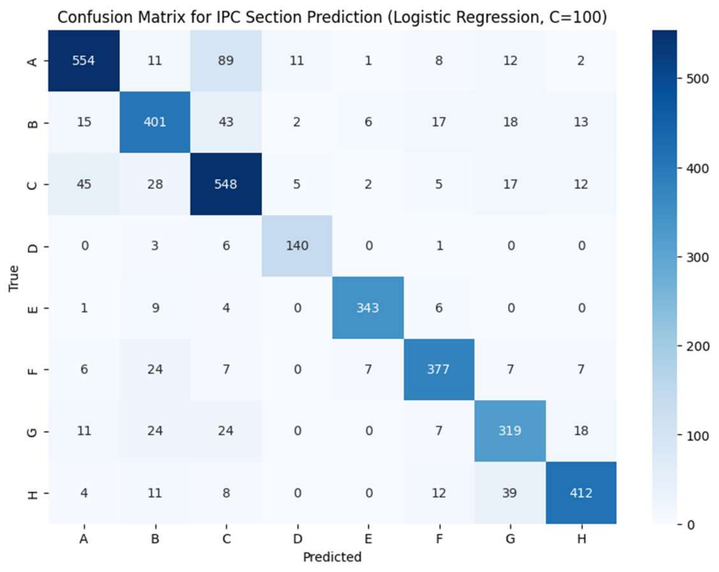


Figure: Confusion matrix for LogisticRegression (c = 100)

The selection of the final model was based on a series of experiments that compared different classifiers using standard evaluation metrics such as **Accuracy**, **Precision**, **Recall**, and **F1-score**. These metrics help assess the model's performance in classifying IPC sections.

Metrics Definitions

- **Accuracy**: Accuracy measures the overall correctness of the model by calculating the ratio of correctly predicted instances (true positives and true negatives) to the total number of predictions. It can be expressed as:

  $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  Where,
  - TP = True Positive
  - TN = True Negative
  - FP = False Positive
  - FN = False Negative

- **Precision**: Precision reflects how many of the instances predicted as positive are actually positive. It is a measure of the model's accuracy in making positive predictions and is defined as:

  $$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall (Sensitivity or True Positive Rate):** Recall measures the ability of the model to correctly identify positive instances out of all actual positive instances. It is expressed as:

  $$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-score**: The F1-score is the harmonic mean of Precision and Recall, providing a balance between the two metrics. It is particularly useful when the model's predictions need to balance both false positives and false negatives. The F1-score is defined as:

  $$\text{F1-score} = \frac{2*(Precision*Recall)}{Precision + Recall}$$

In general, for good performance, Accuracy, Precision, Recall, and F1-score should be high. The following summarizes the performance of the kNN classifier across 8 IPC sections (A–H), with detailed observations for accuracy, precision, recall, and F1-score:

- **Accuracy:** Overall accuracy for kNN was **81.3%**, indicating a generally good fit.
- **Precision:**
  - Highest precision for IPC sections **D** (95.3%), **E** (94.2%), and **F** (87.1%).
  - Lowest precision for section **B** (74.5%).

- **Recall:**
  - Best recall for sections **E** (94.7%), **D** (85.1%), and **H** (85.1%).
  - Lower recall for section **C** (73.9%).
- **F1-score:**
  - Highest F1-score for sections **E** (94.4%), **D** (89.9%), and **F** (85.6%).
  - Lower F1-score for section **B** (76.7%).

IPC Section	Accuracy	Precision	Recall	F1-Score
A	81.3%	77.3%	81.6%	79.4%
B	74.5%	74.5%	79.0%	76.7%
C	77.5%	77.5%	73.9%	75.7%
D	95.3%	95.3%	85.1%	89.9%
E	94.2%	94.2%	94.7%	94.4%
F	87.1%	87.1%	84.2%	85.6%
G	77.4%	77.4%	75.3%	76.3%
H	85.3%	85.3%	85.1%	84.3%

Table: Summary of kNN Classifier performance

Conclusion: In general, with all of the reported results, the model performance is always higher than 70% for all of the metrics.

Below is the comparison of the three different classifiers:

Classifier	kNN (n=3)	LogisticRegression (C=100)	SVM (kernal = poly)
Accuracy	0.813	0.836	0.836
Prec, A	0.773	0.805	0.798
Recal, A	0.816	0.871	0.867
F1, A	0.794	0.837	0.831
Prec, B	0.745	0.778	0.769
Recal, B	0.790	0.784	0.786
F1, B	0.767	0.782	0.777
Prec, C	0.775	0.828	0.831
Recal, C	0.739	0.752	0.751
F1, C	0.757	0.788	0.789
Prec, D	0.953	0.933	0.933
Recal, D	0.851	0.886	0.886
F1, D	0.899	0.909	0.909
Prec, E	0.942	0.945	0.942
Recal, E	0.947	0955	0.961
F1, E	0.944	0.950	0.951
Prec, F	0.871	0.867	0.874
Recal, F	0.842	0.871	0.872
F1, F	0.856	0.869	0.873
Prec, G	0.774	0.792	0.799
Recal, G	0.753	0.774	0.771
F1, G	0.763	0.783	0.785
Prec, H	0.853	0.848	0.856
Recal, H	0.851	0.888	0.895
F1, H	0.843	0.867	0.875

Table: Metrics of different classifiers

Conclusion: Based on the summary metrics, SVM performed the best, followed by Logistic Regression, and kNN had the lowest performance. However, due to resource limitations, we recommend using Logistic Regression for model development.

2.3 IPC Section Classification:

We used a logistic regression model to predict IPC sections based on patent claims embeddings generated by the PatentSBERTa model. The key steps involved were:

Data Preparation

Dataset: We used the cleaned dataset train_dataset.csv, containing patent claims and IPC codes. Due to limited resources, we used a smaller number of dataset to show our intended project.

Preprocessing:

- Rows with missing or empty claims were removed to ensure valid inputs.

```
df = df[df['claims'].notna() & (df['claim1s'].str.strip() != '')]
```

- The first letter of the IPC code was extracted as the IPC section.

```
df['ipc_section'] = df['ipc'].apply(lambda x: x[0] if pd.notna(x) and len(x) > 0 else '')
```

- We used **LabelEncoder** to convert the IPC sections into numeric labels for classification.

```
label_encoder_section = LabelEncoder()
df['encoded_section'] = label_encoder_section.fit_transform(df['ipc_section'])
```

Encoding Patent Claims

- The data is split into training and testing sets to evaluate the model's performance on unseen data. The split ensures 20% of the data is reserved for testing.

```
X_train, X_test, y_train, y_test = train_test_split(df['claims'], df['encoded_section'], test_size=0.2,
random_state=42)
```

- The **PatentSBERTa** model was used to encode the patent claims into embeddings, providing semantic representations of the text that serve as input features for the classifier. The text is encoded into numerical vectors that encode linguistic patterns, relationships, and domain-specific terminology. By using these embeddings, the classifier can understand technical jargon, invention structure, and the relationships between different terms, making it more effective at identifying the relevant IPC sections.

```
model = SentenceTransformer('AI-Growth-Lab/PatentSBERTa')
X_train_embeddings = model.encode(X_train.tolist())
X_test_embeddings = model.encode(X_test.tolist())
```

Classifier: Logistic Regression

- We trained a logistic regression model to predict the encoded IPC sections using the patent claim embeddings.

```
classifier = LogisticRegression(max_iter=1000)
```

- **Cross-Validation**: We employed 5-fold **StratifiedKFold** cross-validation to evaluate the model's performance across different training/test splits. The metrics observed for each fold include **Accuracy**, **Precision**, **Recall**, and **F1-score**.

```
skf = StratifiedKFold(n_splits=5)
```

Results

- The performance of the model was evaluated over 5 cross-validation folds and visualized for each fold: **Accuracy, Precision, Recall, F1-score**
- The results for each fold are plotted to track the model's performance over multiple simulations.
- **Final Evaluation**
  - After cross-validation, the model was trained on the full training set and evaluated on the test set.
    - **Test Accuracy**: 0.9376
    - **Test Precision**: 0.9501
    - **Test Recall**: 0.9376
    - **Test F1-score**: 0.9396

The **classification report** provides a detailed breakdown of performance for each IPC section. A **confusion matrix** visualizes the final predictions, showing where the model made correct and incorrect classifications for each IPC section.

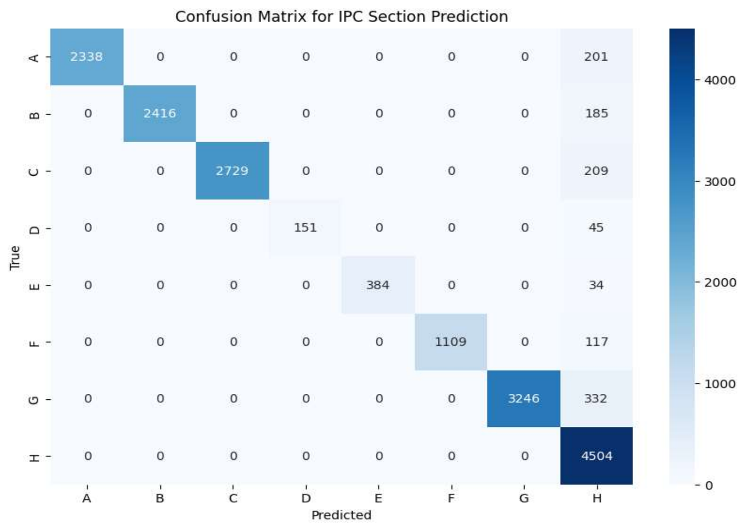


Figure: Confusion matrix for IPC section prediction

The model demonstrates high performance across all IPC sections, with particularly strong results for sections A, B, C, and G. IPC section H shows a lower precision but compensates with a high recall, indicating the model's tendency to correctly classify most samples in section H, even at the cost of some false positives.

Save the Trained Model

- The trained classifier and label encoder are saved for future use, allowing quick loading and inference without retraining.

  ```
  joblib.dump(classifier, classifier_file_path)
  joblib.dump(label_encoder_section, label_encoder_file_path)
  ```

2.4 Prior Art Generation

For prior art searches, we want to automate the process of searching through similar claims or patents. We have encoded a test dataset of claims into a large corpus of embeddings, these are numerical representations of the words, sentences, and relationships in text. Then we used cosine similarity and cosine distance computations to determine how closely related or similar different claims are, based on the semantic context.

Cosine similarity measures the cosine of the angle between two vectors in an n-dimensional space, resulting in a similarity score between -1 and 1. A score of 1 means the vectors are identical (in the direction of the angle), 0 means they are orthogonal (no similarity), and -1 indicates they are diametrically opposite.

Cosine similarity is used to compare the embeddings of different patent claims to find similarities between them.

You can use it to check how closely related two patent claims are, which is crucial in tasks like patent classification, prior art detection, or searching for relevant patents.

For a large patent corpus, such as EPO patent database, you can use cosine similarity to compare an input patent claim with every other patent claim in the corpus. This allows for tasks:

Similarity search: Identifying patents with claims most similar to the input.
Prior art search: Detecting existing patents that are conceptually close to new claims.
Patent clustering: Grouping patents based on the similarity of their claims.

2.5 IPC Section/Class/Subclass Classifier

We then extended the code to classify the class and subclass using **PatentBERTa** and **LogisticRegression.** The approach involved the following key steps:

2.5.1 Data Loading and Preprocessing

Data Loading:

- The code loads the dataset train_dataset.csv, assumed to have undergone some initial preprocessing. This CSV file includes columns like ipc (IPC code), claims (patent claims), and publication_number.

- It filters out rows where claims are empty to ensure model training on non-null data only.

IPC Code Extraction:

- IPC codes are segmented into sections (first character), classes (first three characters), and subclasses (first four characters) to build multi-level classification.

- Label Encoding: Converts the IPC sections, classes, and subclasses into numerical labels required by the logistic regression classifier.

Data Splitting:

- The dataset is split into training and testing subsets to evaluate model generalizability.

Data Saving:

- Filtered and cleaned data are saved in separate files for reuse.

2.5.2 Training Models

Model Architecture:

- Logistic Regression is chosen for its interpretability and efficient performance with embeddings. The embeddings from PatentSBERTa transform the textual claims into numerical features that are fed into the classifier.

Embedding Generation:

- PatentSBERTa encodes each claim as a dense vector, capturing the semantic meaning. These embeddings are used as input to the logistic regression model.

Cross-Validation:

- Stratified K-Fold cross-validation is used to ensure a balanced distribution of IPC labels across each fold. It calculates metrics for each fold and averages them to determine model robustness.

- Checkpoints are saved after each fold, allowing training to resume from the last completed fold in case of interruptions.

Training for Section, Class, and Subclass Prediction:

- Section Prediction: Uses the first character of the IPC code.
- Class Prediction: Uses the first three characters.
- Subclass Prediction: Uses the first four characters.
- Each level of prediction requires training a separate model due to differing label granularities.

2.5.3 Evaluation of Model Performance

Metrics Calculation:

- For each model (section, class, subclass), the code calculates accuracy, precision, recall, and F1-score to understand the performance from various perspectives.

- A confusion matrix is plotted for each classification level, providing insight into how well the model distinguishes between different IPC classes.

Visualization:

- Training metrics (accuracy, precision, recall, F1-score) are plotted across cross-validation folds, simulating epochs, to visualize model convergence and performance stability.

2.5.4 Saving Models and Checkpoints

- After each cross-validation fold, checkpoints are saved for:
  - The classifier (ipc_section_classifier.pkl, etc.)
  - Metrics lists
  - The current fold number (to resume from the last completed fold)
- The final models and label encoders for each IPC level are saved, allowing reuse without retraining.

2.5.5 Final Output and Result Saving

- Evaluation results for each model level (section, class, subclass) are saved in predicted_eval_dataset.csv or similar files.

- These files contain the publication number, claims, true IPC code, predicted IPC code, and whether the prediction was accurate.

2.6 Multi-label IPC Section Prediction and Evaluation

We implemented a multi-label classification system to predict IPC sections from patent claims using the **PatentSBERTa** model. The approach involved the following key steps:

2.6.1 Data Preparation

- **Train and Evaluation Datasets:** The training dataset (train_dataset.csv) and evaluation dataset (eval_dataset.csv) were prepared with claims and IPC section lists.

- IPC sections were extracted by splitting and cleaning the full IPC codes, and a **MultiLabelBinarizer** was applied to convert these into a multi-label format for classification.

2.6.2 Model Architecture

- **Embedding Patent Claims:** We used the **PatentSBERTa** model to convert patent claims into vector embeddings, which served as input for the neural network.

- **Neural Network Architecture:**
    - A fully connected neural network (also known as a multi-layer perceptron) is designed to perform the multi-label classification task. The network consists of the following layers:

        - **Input Layer**: The input dimension matches the length of the vector embeddings generated by PatentSBERTa.

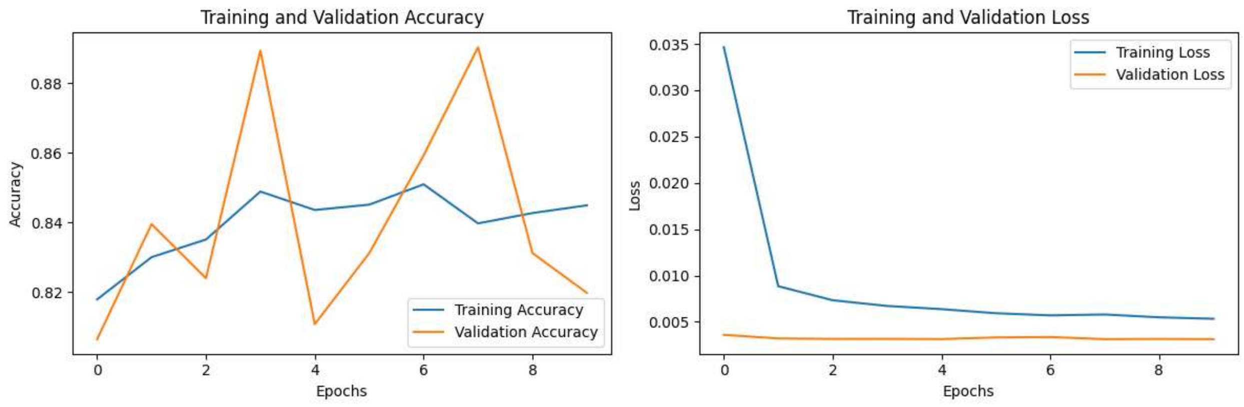
        - **Dense Layers**: Two fully connected dense layers are added, each with 512 and 256 neurons respectively, and ReLU activation functions are used to introduce non-linearity. These layers aim to learn abstract features from the claim embeddings.
        - **Dropout Layers**: To prevent overfitting during training, dropout layers with a dropout rate of 0.3 are placed after each dense layer.

        - **Output Layer**: The final output layer has neurons corresponding to the number of IPC sections (multi-labels). A sigmoid activation function is used to generate probabilities for each section, as multi-label classification requires independent binary predictions for each label.

        - **Training Process:** The model was trained with binary cross-entropy loss and Adam optimizer. Checkpoints were saved after each epoch to ensure recovery in case of failure. We trained the model for 10 epochs.

- **Evaluation and Prediction:**
    - **Evaluation:**
        - The trained model was evaluated on the test dataset using accuracy and loss metrics.
        - We predicted IPC sections using the model and compared them with the actual sections.
- **Performance Metrics:**
        - The model performance was measured based on true predictions (all predicted IPC sections match actual ones), false predictions (no match), and partially correct predictions (half or more sections matched).

Below are the performance metrics of the multilabel model prediction training.



- **Results**
    - **Results File:** The predicted IPC sections, along with actual IPC sections, were saved in predicted_eval_results.csv, which included a column indicating the prediction status: True, False, or True/False.

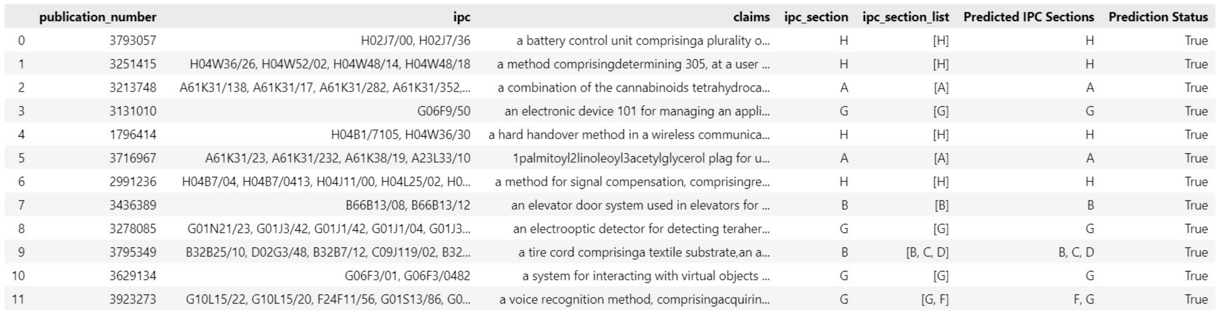
    - **Accuracy Analysis:** We observed the total counts of True, False, and True/False predictions:
        - Total **True** Predictions: true_count (true prediction)
        - Total **False** Predictions: false_count (false prediction)
        - Total **True/False** Predictions: true_false_count (partial correct prediction)

The model was evaluated on the multi-label IPC section prediction task, yielding the following results:

- **Accuracy:** 81.98%
- **Loss:** 0.0031
- **Total True Predictions:** 24,789
- **Total False Predictions:** 211
- **Total True/False Predictions:** 0

○

The Figure below shows the output of the multilabel prediction



	publication_number	ipc	claims	ipc_section	ipc_section_list	Predicted IPC Sections	Prediction Status
0	3793057	H02J7/00, H02J7/36	a battery control unit comprisinga plurality o...	H	[H]	H	True
1	3251415	H04W36/26, H04W52/02, H04W48/14, H04W48/18	a method comprisingdetermining 305, at a user ...	H	[H]	H	True
2	3213748	A61K31/138, A61K31/17, A61K31/282, A61K31/352,...	a combination of the cannabinoids tetrahydroca...	A	[A]	A	True
3	3131010	G06F9/50	an electronic device 101 for managing an appli...	G	[G]	G	True
4	1796414	H04B1/7105, H04W36/30	a hard handover method in a wireless communica...	H	[H]	H	True
5	3716967	A61K31/23, A61K31/232, A61K38/19, A23L33/10	1palmitoyl2linoleoyl3acetylglycerol plag for u...	A	[A]	A	True
6	2991236	H04B7/04, H04B7/0413, H04J11/00, H04L25/02, H0...	a method for signal compensation, comprisingre...	H	[H]	H	True
7	3436389	B66B13/08, B66B13/12	an elevator door system used in elevators for ...	B	[B]	B	True
8	3278085	G01N21/23, G01J3/42, G01J1/42, G01J1/04, G01J3...	an electrooptic detector for detecting teraher...	G	[G]	G	True
9	3795349	B32B25/10, D02G3/48, B32B7/12, C09J119/02, B32...	a tire cord comprisinga textile substrate,an a...	B	[B, C, D]	B, C, D	True
10	3629134	G06F3/01, G06F3/0482	a system for interacting with virtual objects ...	G	[G]	G	True
11	3923273	G10L15/22, G10L15/20, F24F11/56, G01S13/86, G0...	a voice recognition method, comprisingacquirin...	G	[G, F]	F, G	True

Figure: Multilabel prediction output

The model consistently predicted the correct IPC sections for most samples, demonstrating high accuracy with minimal false predictions. This was intended to showcase the feasibility of multi-label IPC prediction, as some patent claims fall under multiple IPC codes. However, due to the complexity of implementation, the limited time available in the competition, and resource constraints, we primarily focused on illustrating the potential of this approach through the source code. The main project implementation was centered around single-label IPC section classification.

Chapter 3: Project Implementation

There are three indispensable stages:

- Stage 1: Extracting the patent data into the appropriate structures from EPO TIP platform in large amounts> 10000. Preprocessing steps are then conducted on the large extracted (clean up the data, randomize the data, etc. For details see Fig 2.1.1).

- Stage 2: Build the model, and experiment with different model architecture. The core model is built to solve the two fundamental tasks in generating the EPO examiner report: performing ipc categorization, and generating the prior arts and their similarity scores.

- Stage 3: Build the interactive interface for the model and the user.

The demonstration code of the project is at Codefest EPO 2024, Confused Electrons - Gen Ai application

An example project implementation is given in the code which will show part of what our intended project is.

Project Implementation Overview

This project comprises three main components aimed at creating a mock patent application submission system, predicting patent classification for examiners, and generating relevancy information for prior art in an automated search report. Due to time constraints, some parts of the project were simplified, and mock data was used where necessary. Below is the detailed breakdown of the project implementation.

Patent Application Submission Portal

The first part of the project involves creating a Patent Application Submission Form using a GUI (Tkinter). This portal is designed for patent applicants to submit their patent applications by providing necessary details such as:

- Applicant information (name, address)
- Inventor information (name, contact details)
- Patent details (title, abstract, claims, description)
- Field of invention

The form also allows applicants to upload supporting documents (e.g., drawings, descriptions) and save the application data into CSV format. This simulates a real-world patent submission portal. The submitted information is saved to a directory based on the patent title for further processing.

Search Report Generation for EPO Examiners

The second code is used to facilitate patent examiners at the European Patent Office (EPO) in predicting International Patent Classification (IPC) sections and generating a mock search report based on the submitted applications.

- **IPC Prediction:** Upon selecting a submitted application, the examiner can predict the IPC section based on the provided patent claims using a pre-trained classifier. The IPC predictions are then saved in a CSV file.
- **Search Report Generation**: Once the IPC predictions are made, the system generates a mock International Search Report (ISR) in PDF format. This report includes:
    - Patent classification information
    - Relevant prior art documents (mock data)
    - Other sections like "Fields Searched" and "Authorized Officers"

However, the "Documents Considered to be Relevant" section currently contains mock references due to limited time. The plan was to integrate prior art relevancy scores generated by the third code into this section.

Prior Art Relevancy Information Generation

The third component focuses on generating prior art relevancy information based on patent claims. This system calculates the cosine similarity between the submitted patent claims and prior art documents, identifying the most relevant prior art references.

- **Patent Embeddings:** Using PatentSBERTa, the system converts the patent claims into embeddings and compares them with existing prior art documents.

- **Cosine Similarity:** The similarity scores help in identifying the most relevant prior art documents.

- **Relevancy Results:** The results, including publication numbers and similarity scores, are generated and saved to a CSV file.

While the relevancy information was successfully generated, integrating this data into the "Documents Considered to be Relevant" section of the search report was not completed due to time constraints. Hence, mock references were used in this section instead of the actual relevancy results from this process.

Chapter 4: Conclusion

The solution proposed demonstrates a comprehensive and innovative approach to automating the generation of EPO search reports using Generative AI models. By providing claim classification, relevant prior art searches, and report generation, the system aims to streamline the process of search report generation.

- Completeness and Transferability: The solution covers the essential stages of the patent examination process: patent classification and prior art generation. It provides a structured approach from data extraction and preprocessing to building models and deploying them into an interactive interface. The system can be run on EPO TIP, as it was built on standard Jupyter Notebooks.

- Effectiveness and Efficiency: The use of advanced NLP models, such as PatentSBERTa, ensures high accuracy in classifying patents according to IPC sections and retrieving relevant prior art. The significant improvements in classification accuracy (81.98%) and reduced time for report generation.

- Design and Usability: The solution is designed with practicality and usability in mind. The workflow—from data extraction to report generation—has been optimized to fit into existing systems at the EPO. The interactive interface built into the system allows examiners to easily interact with the model, improving usability and facilitating a smoother integration into current workflows.

- Creativity and Innovation: The integration of Generative AI for generating EPO search reports represents a novel application of AI in the patent field. Leveraging machine learning for both classification and prior art generation, alongside an interactive platform for users. This project offers a new pathway for improving accuracy, speed, and overall effectiveness in the Search Report Generation process.

In conclusion, the proposed solution meets the evaluation criteria by delivering an effective, and innovative system that addresses key challenges in the patent process while being highly compatible with other datasets from EPO.