Travail Encadré de Recherche

UNIVERSITÉ PARIS-SACLAY

FACULTY OF SCIENCES - MASTER IN DATA SCIENCE

Preparation of a challenge in Retrieval Augmented Generation on technical texts.

Supervisor: Kim Gerdes
Academic Year 2023-2024
Pablo Mollá Chárlez

Contents

1 Project and Objectives

In early January 2024, I embarked on an engaging project with Qatent as part of my Master's in Data Science program under the guidance of Professor Kim Gerdes. This opportunity arose from a collaboration aimed at enhancing patent analytics through the use of data science techniques. Alongside three fellow students and PhD students, we formed a diverse team tasked with utilizing the PatentMatch dataset as a baseline. Through various modifications—including filtering, analysis, and data visualization—and creating new subdatasets, we developed a data challenge hosted on CodaBench for students in the Information Retrieval course. We applied machine learning algorithms such as BM25, TF-IDF, and transformer models (BERT, SBERT) to make predictions on citations, paragraphs, claims, and various features related to patent matters.

1.1 Purpose of the Report

The primary purpose of this report is to present the context, document the tasks performed by myself, the modifications made to the PatentMatch dataset, and the insights gained throughout the duration of the project. This report aims to showcase the practical applications of data science in patent law and to illustrate the project's contributions to the field of patent analytics and to the Information Retrieval course.

2 Fundamentals of Patent Law

2.1 Definition of a Patent

Patents are like official certificates given by an institution that grant inventors the exclusive right to make, use, sell, and import an invention for a certain period of time, usually 20 years. Think of them as protective shields for inventions. These inventions can be new gadgets, chemical formulas, processes, or any new and useful improvement of an existing invention. There are multiple reasons to why patents are important and interesting:

- protection for inventors, encourages innovation, economic growth, revenue generation, sharing knowledge.

The duration of patent protection varies by region. In general, a patent typically lasts for 20 years from the filing date of the application. For instance, the European Patent Office (EPO) and the United States Patent and Trademark Office (USPTO) patents have a standard term of 20 years from the date of filing the application. It is essential to understand that to keep a patent in force, maintenance fees may be required, and these fees can vary between jurisdictions. It's also worth noting that the scope of patent protection is regional. A patent granted by the EPO is valid in European member states. When seeking patent protection in multiple regions, an applicant can file an international patent application under the World Intellectual Property Office (WIPO).

2.2 Structure of a European Patent

A European patent is a legal document protecting an invention in multiple European countries and is administered by the European Patent Office. The typical structure of a European patent document includes the following components: [1]

- **Date of Publication:** The date on which the patent document is officially published by the European Patent Office.
- **Application Number:** A unique identifier assigned to the patent application when it is filed.
- **Date of Filing:** The official date on which the patent application was submitted to the EPO.
- **Designated Contracting States:** The countries in which patent protection is sought, selected from the member states of the European Patent Convention (EPC).
- **International Patent Classification (IPC):** Codes assigned to the patent, indicating the technical fields to which the invention relates.
- **Applicant & Inventor:** The names of the applicant (the entity applying for the patent) and the inventor(s) (the person(s) who created the invention).

[1] See the Appendix (5.1) for an example of a European patent.

- **Abstract:** A brief summary of the invention and its potential applications.
- **Description:**
  - *Field of Invention:* The technical field to which the invention belongs.
  - *Background Art:* Prior art and existing technologies relevant to the invention, highlighting the problems addressed.
  - *Summary of Invention:* An overview of what the invention is and how it addresses the problems identified in the background.
  - *Description of Drawings:* Explanation of the figures and drawings included in the patent document.
  - *Detailed Description:* A comprehensive, detailed explanation of the invention, often supported by examples and embodiments.
- **Claims:** The most crucial part of the patent, defining the scope of protection conferred by the patent. Each claim specifies particular technical features of the invention.
- **Figures:** Drawings and diagrams that illustrate the invention, essential for understanding the invention.
- **International Search Report:**
  - *Documents considered to be relevant:* Lists prior art documents that were taken into consideration during the search.
  - *References Cited in the Description:* Additional documents cited by the inventor or applicant that are relevant to the patentability of the invention.

This structure ensures that the patent document is comprehensive and provides all necessary details to understand the invention fully, assess its novelty, and determine the scope of protection it seeks.

2.3 Citations

When filing a patent, it's strategic to reference relevant prior art to show that you're aware of the existing technologies in the field. Putting too much focus on existing inventions (prior art) could unintentionally make your own invention seem less unique or innovative, or give the impression that it's just a minor variation.

The aim should be to carefully reference the prior art that your invention improves upon or is distinct from. These citations assess the novelty, inventive step, and patentability of the invention. Here we discuss three specific types of citations often encountered in patent documents: X, A, and Y citations.

2.3.1 Types of Citations

$\boxed{\text{X}}$ **Definition** An X citation indicates highly relevant prior art that challenges the novelty or non-obviousness of the claimed invention.

$\boxed{\text{A}}$ **Definition:** An A citation refers to documents that provide relevant technological background but do not directly relate to the novelty of the invention.

$\boxed{\text{Y}}$ **Definition:** Y citations are relevant to the novelty or inventive step but do not conclusively challenge the patentability.

In the realm of patent analytics and research, the utilization of comprehensive datasets is crucial for developing tools and methods that enhance our understanding and processing of patent documents.

Patent examiners need to solve a complex information retrieval task when they assess the novelty and inventive step of claims made in a patent application. Given a claim, they search for prior art, which comprises all relevant publicly available information. This time-consuming task requires a deep understanding of the respective technical domain and the patent-domain-specific language.

In response to the challenge of identifying relevant historical inventions, a specialized dataset named PatentMatch was developed in 2020 to aid in the computerized identification of prior art. This dataset was crafted to support supervised machine learning approaches, allowing for the more refined training of algorithms in this domain.

3 PatentMatch: A Dataset for Matching Patent Claims & Prior Art

The PatentMatch dataset, as detailed on its GitHub repository, is a structured collection of patent texts and associated metadata, designed to support the matching of patent claims with prior art, with entries dating back to 1978.

This compilation, structured originally in XML format, includes matched sets comprising patent application claims and associated textual sections of varying relevance from referenced patents processed by the European Patent Office. A key addition post-2012 is the inclusion of search reports that assist in evaluating the uniqueness and innovative progress of the patent applications.

These sets have been meticulously annotated by the expert patent examiners at the EPO, designating each pair with a tag/category that reflects the semantic relationship between the claim and the text excerpt. The tags A, X and Y, as previously mentioned, serve to indicate the potential impact of the text on the originality of the patent claim in question.

3.1 Original Dataset & Parsing Script

The script 0_parse.py is developped to read XML files sourced from the EPO's bulk data sets designed for text analytics, which can be found at the EPO's data website. Its primary function is to meticulously extract and structure all relevant information from these XML files.

In my role, it was imperative to gain a thorough understanding of how this extraction process was carried out by the original script. I encountered the need to modify the baseline script to rectify inconsistencies and ensure the extraction process aligned with our specific project objectives. This involved filtering out erroneous data and selectively parsing data that was pertinent to our study — specifically, descriptions in English and data falling within the designated timeframe, which included citations and patent applications from 2010 to the year 2020.

To improve data quality, I performed cleansing operations on the fields and employed regular expressions, which proved essential for retrieving certain pieces of information that the initial functions had overlooked. The ultimate challenge was to successfully apply this refined parsing process to the entirety of the 28 out of 40 XML files (each approximately 5GB in size), to effectively handle and process the complete filtered dataset of 73 out of 200GB. This comprehensive processing [2] was a pivotal step in preparing the data for subsequent stages of our project.

[2]See the Appendix (7.2) for the multiple functions.

3.2 Docker & ElasticSearch

After completing the initial parsing of the data using the 0_parse.py script, we recognized the necessity to enhance our data management system. To achieve this, we decided to utilize Docker.

- Docker is a popular open-source platform that simplifies the creation, deployment, and running of applications using containerization technology.
- Docker containers wrap software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries – anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.

One of our team members took the initiative to use Docker on the 'calcul' server offered by the Laboratoire Interdisciplinaire des Sciences Numériques (LISN) of the Université Paris-Saclay, provided by our professor. This strategic move was essential for managing the extensive datasets effectively.

Using Docker, we dowloaded the Elasticsearch's official image to create the corresponding container and handle the large volumes of text data extracted by our parsing script.

- Elasticsearch, a powerful open-source search and analytics engine, is well-suited for navigating vast amounts of textual data.
- ElasticVue [3] , a web-based client that interfaces with Elasticsearch to provide a more accessible view of the indexed data, enhancing our ability to query and manipulate the information effectively.

The dataset retrieved with 0_parse.py was organized into 2 distinct subdatasets which represent 1428046 patent applications ($\approx$ 72.9GB) and 1287385 patent citations ($\approx$ 310MB).

1. The first segment, denoted 'ep_patent_applications', encompasses all patent applications, detailing elements like the abstract, title, claims, and others, including their citation identifiers.
2. On the other hand, the second segment, named 'ep_patents_citation' houses each citation identifier alongside its relevant citation data. These two segments are interlinked by the shared citation identifiers, allowing for integrated analysis between patent submissions and their references.

This approach not only streamlined our data processing workflow but also enhanced the robustness and scalability of our data management system, ensuring efficient handling and retrieval of information as needed for our project's success.

3.3 Citing, Cited & Mapping Datasets

One of my fellow comrades was in charge of further segmenting the 'ep_patent_applications' and 'ep_patent_citations' datasets from ElasticSearch, extracting and organizing them into four more specialized JSON datasets.

These datasets are crafted by extracting content information from the filtered and indexed data previously stored in Elasticsearch. The structuring into JSON format ensures that each piece of content is meticulously referenced by IDs, facilitating straightforward access and manipulation.

It's important to emphasize that within these datasets, we selectively focused on citing documents within the "A" and "G" classifications of the International Patent Classification (IPC) system. The "A" classification considers a spectrum of patents belonging to Human Necessities. On the other hand, the "G" classification covers Physics, incorporating patents that frequently involve cutting-edge technologies and intricate physical principles. Furthermore, we honed in on patents designated as "A", "X", or "Y"

[3]See the Appendix (7.3) for a ElasticVue preview of the collected data.

within the search reports crafted by the patent examiners, ensuring a targeted approach for our analysis.

The aforementioned datasets are the following:

- content_citing.json (10,779 documents): This dataset comprises the content information of the patents that are citing other patents. It focuses on the textual content of the citing patents.
- mapping_citing_cited.json (10,779 documents): This dataset creates mappings between the citing patents and the patents they cite, establishing a direct reference system between different documents.
- content_cited.json (12,195 documents): This dataset contains the content of the patents that have been cited by other patents, capturing the essential information of the patents referenced in the citations.
- content_uncited.json (10,000 documents): This includes patents that have not been cited by others in the dataset, providing a baseline of patents for comparative purposes.

With these datasets, we were equipped to begin the necessary preprocessing steps to more directly analyze and understand the relationships within the data, task in which a colleague and I were solely responsible for.

Specifically, we aimed to identify and extract relevant paragraphs and claims from the cited patents, focusing on those references made by citing patents. This process is crucial for our primary objective: predicting the relationships and references between patents based on their content and citation patterns. To accomplish such task, we created the Q7.py script and its corresponding Q7.json dataset.

3.4 Q7 Dataset

The script Q7.py was created with the objective of synthesizing data from three out of the four previous JSON datasets: 'content_citing.json', 'mapping_citing_cited.json', and 'content_cited.json'. The goal was to perform a refined intersection of this information into a new JSON structure that clearly delineates all pertinent details from individual patents. [4]

In assembling this consolidated dataset, we aimed to capture specific fields: the citing patent ID ('ID_Citing'), the category of the citing patent ('Category_Citing'), the text of its claims ('Claims_Text'), the cited patent ID ('ID_Cited'), the references cited within the search report ('References_Cited'), and the cited content ('Content_Cited').

Our task was multifaceted and complex, particularly when it came to extracting and cleanly formatting the content from the citations as noted by the examiners in the European Search Reports. These examiners each had their unique style of noting down the interrelation between the citing patents and their citations.

For instance, a patent's information such as its abstract, description, figures, and claims content needed to be pulled and cleansed from our datasets [5]. We then had to carefully link this content with the relevant portions from the cited patents, as determined by the examiner.

This process involved extracting elements such as the abstract, specific paragraphs, and detailed descriptions of cited figures, a task that required a versatile approach due to the diversity of referencing styles employed by the examiners.

[4] See the Appendix (7.4) for an illustration of the expected Q7.json Dataset.

[5] See Appendix (7.4) for a content patent instance.

Regular expressions played a key role in our extraction process, as we faced a multitude of referencing formats within the citations. Our challenge was to design regular expressions that were broad enough to match any potential structure yet precise enough to retrieve only the relevant. For example, a citation might reference an entire document, specific claims, a range of claims, particular paragraphs, or even just figures and tables such as:

- The whole document; claims 5, 7, 8; claim 6; figures 1, 6, 7, 8, 9.
- the whole document; claims 7-10; claim 6; figures 1-6, 7, 8, 9.
- In particular, paragraphs 0013-0018, 0034, 0037-0039, 0045-0049; figures 2, 4.
- whole document; paragraph [0092] - paragraph [0095]; claim 4; example 6; tables 1, 2.
- Figures 2, 3, 3B, 3C, 4, 6, 7; figures 1-9; paragraph [0001] - paragraph [0009]; paragraph [0017] - paragraph [0026].

This thorough and detailed extraction process lead us to develop numerous functions [6] to tackle each situation and allowed us to generate a dataset perfectly suited for our ultimate goal: to directly ascertain which paragraphs and claims from a cited patent are being referenced by a citing patent, thereby facilitating accurate predictions of patent citations.

3.5 Training and Test Datasets

Once the Q7.json dataset was compiled and finalized, the next phase of our project involved dividing this comprehensive dataset into 4 more targeted subsets. This process was essential for creating distinct training and testing sets tailored for the Codabench challenge. Here's a breakdown of how the dataset was segmented:

- cleaned_content_citing.json - This dataset, containing 7,831 documents, was curated to include only the cleaned content from the citing patents. This meant removing any irrelevant data, ensuring the dataset was primed for analysis.
- citation_Train.json - With 8,860 documents, this file became the training dataset. It includes a selection of patent data specifically chosen to train the machine learning models that participants would develop and refine during the challenge.
- citation_Test.json - Comprising 1,000 documents, this smaller subset serves as the testing dataset. It is utilized to evaluate the performance of the models trained using the citation_Train.json, providing a benchmark for the algorithms developed.
- cleaned_content_uncited.json - This dataset is somewhat unique as it contains a mix of 8,834 documents, plus an additional 8,000, all of which represent patents that weren't cited by others in the dataset. It serves a crucial role in the challenge, used as negative examples in training machine learning models.

The creation of these four datasets marked the completion of the preparation phase for the Codabench challenge. With training and testing datasets ready, participants could engage in the challenge with datasets designed to simulate real-world patent analysis scenarios, ultimately testing their models' abilities to predict and understand patent citations. This structured approach to dataset preparation was pivotal in ensuring that the Codabench challenge would be both comprehensive and representative of actual tasks faced by those working in the field of patent analytics.

[6] See Appendix (7.5) for detailed functions.

4 Dataset RoadMap

In the culminating steps of our project, we refined a significant corpus of patent data into a suite of datasets, each designed with a focused objective to advance our Information Retrieval capabilities for the Codabench challenge. Beginning with the comprehensive raw data from EPO patent filings, we applied a meticulous parsing technique, employing a script to transform this vast information into a more manageable format.

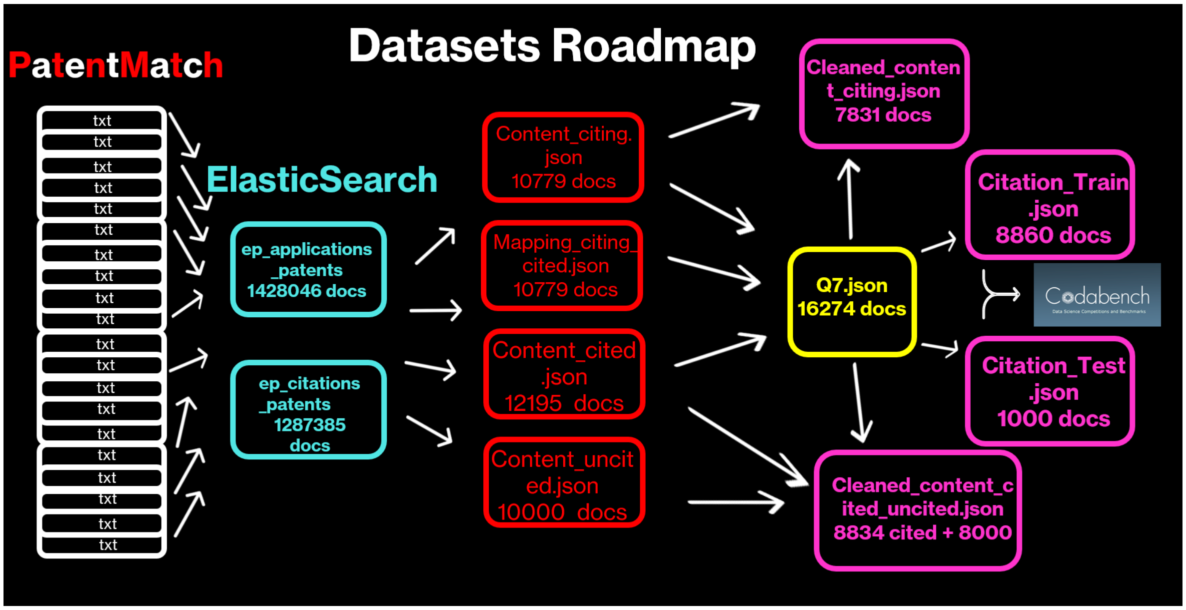
From the outset, our data journey was steered by ElasticSearch, serving as the engine for organizing over a million documents into two primary collections: one for patent applications and the other for patent citations.

These initial datasets were then expertly distilled, thanks to a dedicated script, into four datasets with a collaborative effort (content_citing.json, mapping_citing_cited.json, content_cited.json and content_uncited.json). These datasets—ranging from content of citing patents to mappings of citations and uncited content, were meticulously filtered ("A" and "G" IPC classifications and "A", "X" or "Y" search report citations) referenced by unique identifiers, ensuring precision in our analysis and bridging both cited and citing.

As we synthesized the data, the challenge laid not just in searching through the IDs and categories but in extracting and formatting the content from a multitude of examiner citations. The diversity of these citations demanded robust regular expressions, among other techniques, to capture the necessary details accurately.

Eventually, this granular approach to data curation culminated in the final assembly of training and testing datasets. The 'Citation_Train.json' and 'Citation_Test.json' datasets emerged from this process, marking the end of our dataset preparation and the beginning of a new chapter of challenge preparation. This meticulous and structured approach to dataset creation has been pivotal, ensuring that our contributions to the Codabench challenge are as comprehensive and authentic as the real-world challenges faced by professionals in patent analytics.

For a detailed look at the transformation from raw data to the finalized datasets used in the Codabench challenge, please refer to the accompanying visual roadmap provided as follows.



5 Paper Research: SBERT

In the context of my TER, I also did some research on transformer models. I delved into the nuances of Sentence-BERT (SBERT), a significant modification of the well-known BERT model, as outlined in the original paper from 2019 titled Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks and presented to my comrades the key concepts. The main distinctions and benefits of SBERT over BERT, focusing on its architecture and application in natural language processing tasks are overviewed:

5.1 Overview of SBERT and BERT

SBERT is an adaptation of the original BERT (Bidirectional Encoder Representations from Transformers) designed to enhance the efficiency and accuracy of generating sentence embeddings. While BERT has set benchmarks in various NLP tasks, its architecture inherently faces challenges concerning efficiency, especially in tasks requiring sentence similarity assessments.

5.2 SBERT's Efficient Improvements

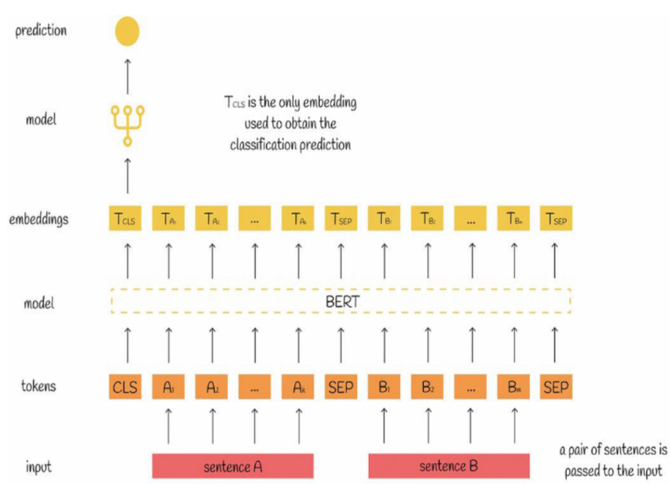
SBERT addresses these challenges by modifying the BERT architecture to produce sentence embeddings more efficiently. Traditional BERT models process sentences in pairs and do not support the reuse of pre-computed sentence embeddings. This approach results in a quadratic computation increase with the number of sentences, making it computationally expensive for large datasets.

In contrast, SBERT allows embeddings to be pre-computed and reused, significantly reducing computation time while maintaining the accuracy of the embeddings. This is achieved by adapting the model to a siamese network structure, which processes each sentence independently, thus supporting tasks like semantic similarity searches more efficiently.

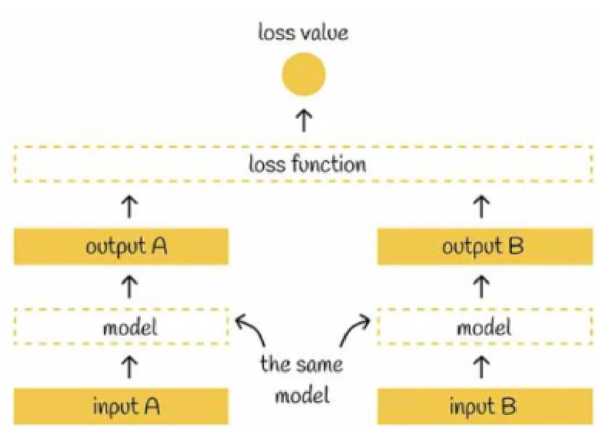
5.3 Architectural Differences

The fundamental architectural change between BERT and SBERT is the move from a cross-encoder to a bi-encoder setup. BERT's cross-encoder evaluates sentence pairs collectively, using the transformer architecture to output a similarity score, which, while accurate, demands substantial computational resources. This setup does not produce standalone sentence embeddings, which limits its applicability for certain types of semantic analyses.

On the other hand, SBERT employs a bi-encoder structure where sentences are processed independently, generating distinct embeddings for each. These embeddings can then be compared using cosine similarity measures, making SBERT much faster and more versatile for applications involving large-scale semantic comparisons.



Cross-Encoder Architecture

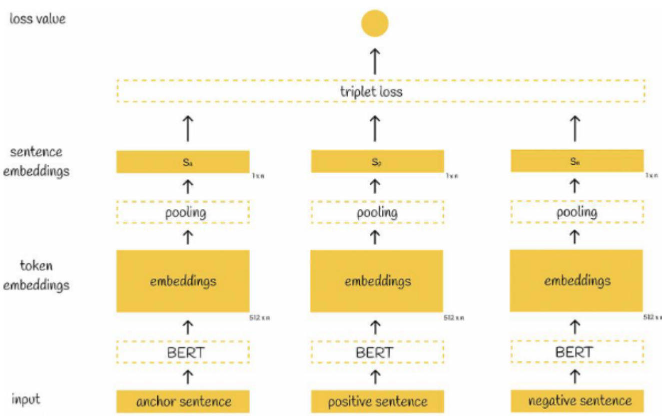


Bi-Encoder Architecture

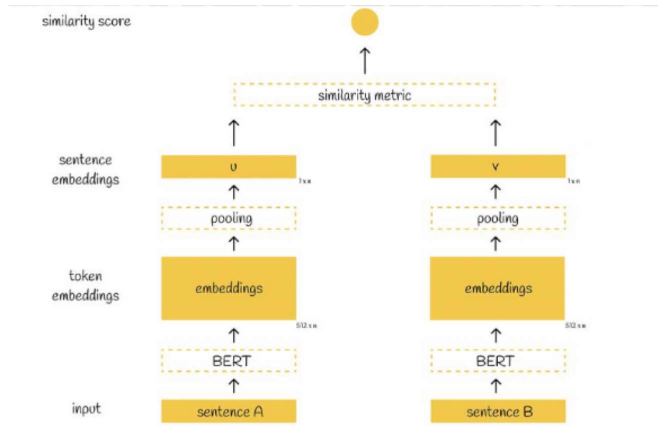
5.4 Semantic Understanding and Training Techniques

SBERT enhances the understanding of semantic meanings in sentences through a specialized training approach. It utilizes siamese and triplet network architectures within its training process, employing a technique known as the triplet loss function. This method involves training on sentence triplets: an anchor sentence, a positive sentence (semantically similar to the anchor), and a negative sentence (semantically different). The model learns to pull the anchor and positive sentences closer in the embedding space while pushing the negative sentence farther away.

This training strategy enables SBERT to achieve a nuanced understanding of sentence semantics, beneficial for tasks requiring precise interpretation of sentence meanings, such as matching questions to answers or identifying similar sentences across documents.



Triple Network



Siamese Network

6 Conclusion

In conclusion, this report has set down the meticulous process of developing a structured and analytical approach to patent data, culminating in the creation of comprehensive datasets for the Codabench challenge. Our journey began with parsing the vast XML formatted patent data, leveraging the prowess of Elasticsearch managed through Docker on the LISN's "calcul" server. A collaborative effort led to the successful preprocessing and extraction of relevant patent information, overcoming challenges posed by diverse citation references.

Subsequent to the assembly of the Q7.json dataset, our team partitioned this data into distinct subsets, creating clean, targeted datasets for training and testing, essential for the machine learning models' development. The resulting datasets not only facilitated a deep dive into the nuances of patent citations but also laid a strong foundation for the Codabench challenge participants to innovate and test their predictive models.

My journey through the complex landscape of patent data has been profoundly educational and immensely enriching. The process of developing structured datasets for the Codabench challenge has endowed me with invaluable insights into patents, data mining techniques, and the application of NLP in machine learning models, particularly BM25, TF-IDF, BERT and SBERT transformers.

The hands-on experience with data preprocessing, information extraction, and the challenges of dealing with complex patent references has sharpened my skills and deepened my understanding of natural language processing and its pivotal role in Information Retrieval.

I am immensely grateful for the collaboration and assistance provided by my peers, whose expertise and shared effort made this project research possible. Their support was instrumental in overcoming the intricate challenges we faced. I am equally thankful for the patience and passionate guidance offered by Professor Kim, whose enthusiasm and deep knowledge in the field were not only inspiring but also illuminating.

7 Appendices

The comprehensive material within the appendix of this report offers just a glimpse of the extensive information that has been compiled. For those who wish to delve into the full breadth of the research and findings, I invite you to visit my personal GitHub repository, where the complete scripts and documentation are accessible.

Throughout my research, I have engaged with a variety of scholarly materials to solidify my understanding of patent law and information retrieval techniques.

While some documents, particularly those central to my research, were read with meticulous attention to detail, others were used to grasp the general concepts and methodologies. This approach allowed me to efficiently allocate my time to the most pertinent subjects while maintaining a broad awareness of the field as a whole.

References

[1] World Intellectual property Office (WIPO), *Patent Draft Manual*, URL : `http://www.ub.edu/centredepatents/pdf/material_referencia/WIPO_Patent_Drafting_Manual_2022.pdf`, 2022.

[2] World Intellectual property Office (WIPO), *Patent Claims Format & Types of Claims*, URL : `https://www.wipo.int/edocs/mdocs/aspac/en/wipo_ip_phl_16/wipo_ip_phl_16_t5.pdf`, 2022.

[3] European Patent Office - EspaceNet, *Patent Search Engine*, URL : `https://worldwide.espacenet.com/patent/`, 2024.

[4] Article on llamaindex.ai, *Running Mixtral 8x7 locally with LlamaIndex and Ollama*, URL : `https://www.llamaindex.ai/blog/running-mixtral-8x7-locally-with-llamaindex-e6cebeabe0ab`, 2023.

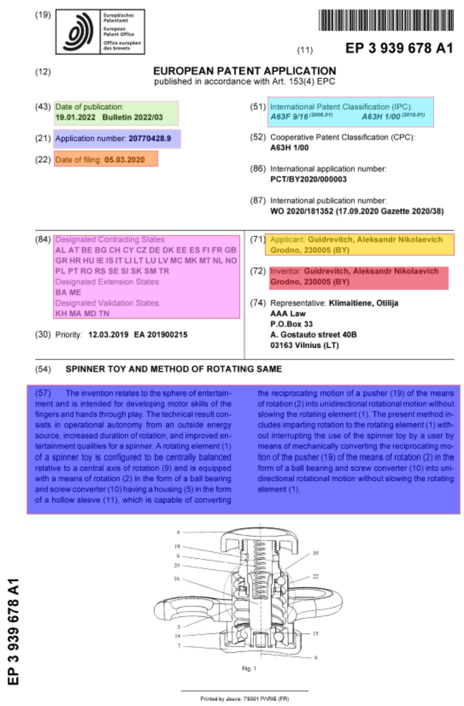
[5] Yunfan Gao, Yun Xiong, *Retrieval-Augmented Generation for Large Language Models: A Survey*, URL : `https://arxiv.org/pdf/2312.10997v4.pdf`, 2024.

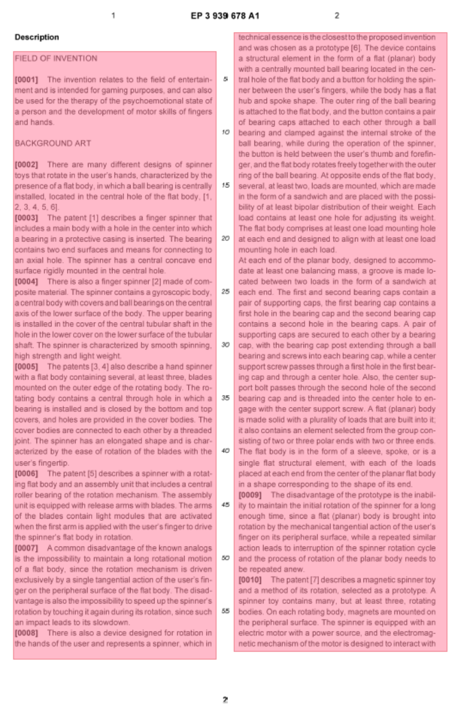
[6] Trey Grainger, Doug Turnbull, Max Irwin, *AI-Powered Search*, URL : `https://www.manning.com/books/ai-powered-search`, 2024.

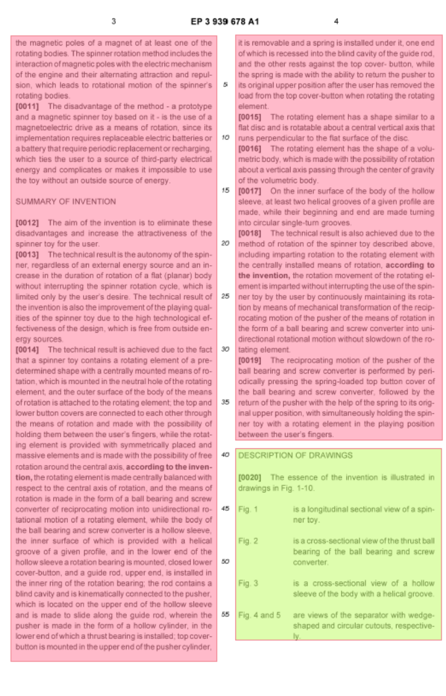
7.1 European Patent Example: Spinner Toy

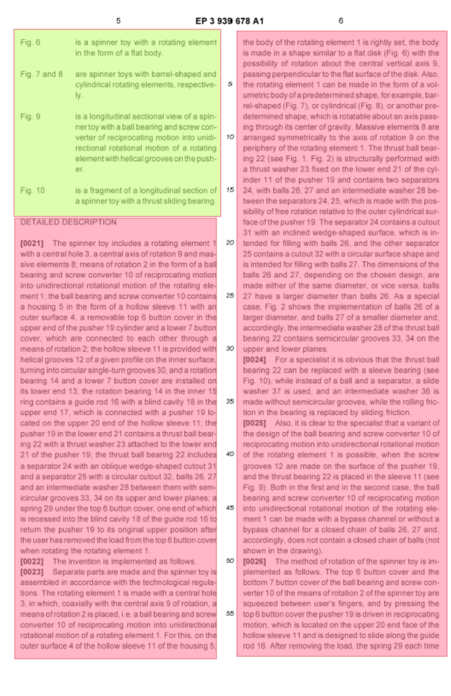
The original document can be found on the official website of the EPO:

- Espacenet Patent Search and Spinner Toy - Patent.









3. The spinner toy according to claim 1, **characterized in that** the rotating element (1) has the shape of a volumetric body, which is configured to rotate about a vertical axis passing through the center of gravity of the volumetric body.

4. The spinner toy according to claim 1, **characterized in that** at least two screw grooves (12) of a given profile are made on the inner surface of the body (5) of the hollow sleeve (11), while their beginning and end are made turning into circular single turn grooves (30).

5. A method of rotating a spinner toy according to claim 1 of the formula, comprising imparting rotation to a rotating element (1) with a centrally installed means of rotation (2), **characterized in that** the rotation movement to the rotating element (1) is imparted without interrupting the use of the spinner toy by the user by continuously maintaining its rotation by means of mechanical transformation of the reciprocating motion of the pusher (19) of the means of rotation (2) in the form of a ball bearing and screw converter (10) into unidirectional rotational motion without slowing down the rotating element (1).

6. The method of rotation of the spinner toy according to claim 5, **characterized in that** the reciprocating motion of the pusher (19) of the ball bearing and screw converter (10) is performed by periodically pressing the spring-loaded top (6) button cover of the ball bearing and screw converter (10), followed by the return of the pusher (19} by the spring (29) to the initial upper position, while simultaneously holding the spinner toy with the rotating element (1) in the play position between the user's fingers.

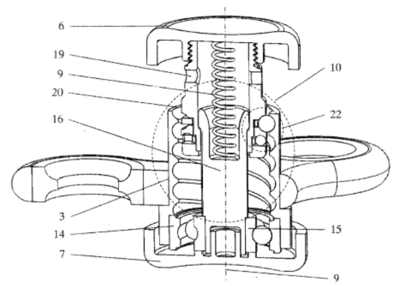


Fig. 1

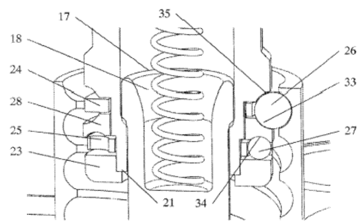


Fig. 2

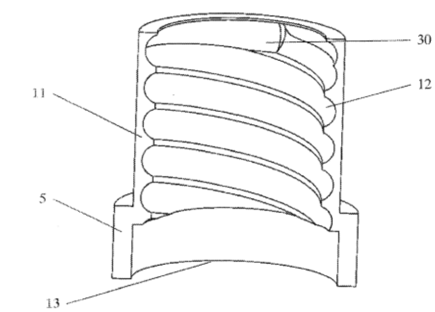
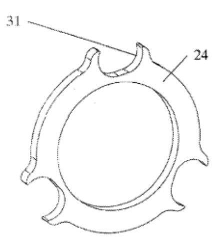


Fig. 3



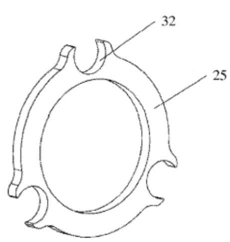


Fig. 4

Fig. 5

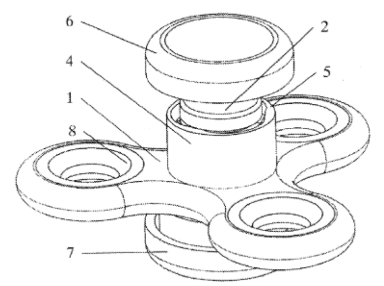
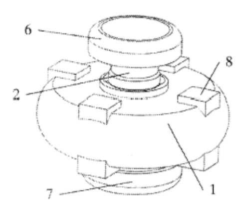


Fig. 6



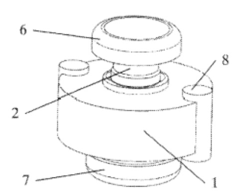


Fig. 7

Fig. 8

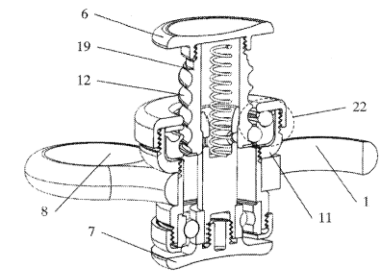


Fig. 9

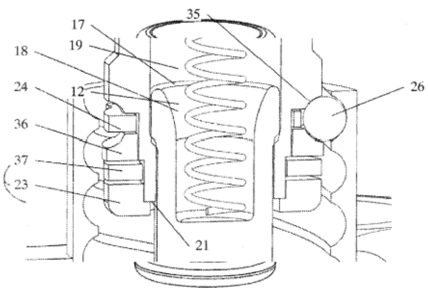


Fig. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/BY 2020/000003

A. CLASSIFICATION OF SUBJECT MATTER
A63F 9/16 (2006.01) A63H1/00 (2019.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
A63F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Espacenet, PatSearch, PAJ, WIPO, USPTO, RUPTO

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5135425 A (ANDREWS MELVIN R; ANDREWS ROGER W) 04.08.1992	1-6
A	US 9914063 B1 (MCCOSKERY MICHAEL SCOTT et al.) 13.03.2018	1-6
A	US 2017326468 A1 (KINMONT JR RICHARD C et al.) 16.11.2017	1-6
A	CN 107638699 A (DONGGUAN DAVANTECH CO LTD) 30.01.2018	1-6

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 May 2020 (14.05.2020)	21 May 2020 (21.05.2020)
Name and mailing address of the ISA/	Authorized officer
Facsimile No.	Telephone No.

Form PCT/ISA/210 (second sheet) (July 1998)

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- TW M551522 **[0028]**
- TW 11112017 **[0028]**
- CN 107754323 **[0028]**
- CN 06032018 **[0028]**
- JP 3212430 B **[0028]**
- JP 07092017 B **[0028]**
- CN 107638699 **[0028]**
- US 9895620 B **[0028]**
- US 99114063 B **[0028]**
- US 5135425 A **[0028]**

7.2 0_parse.py

Some important functions modified to index all the dataset in ElasticSearch.

```
def from_citationIDs_to_application_number(line):
    pattern_patcit = re.compile('patcit dnum="[A-Z]{2}[A-Z\d]+"')
    result_patents = [elem[13:-1] for elem in pattern_patcit.findall(line)]
    return result_patents

def extract_citationIDs(application_identifier, line):
    words = line.split("\t")[7].split(" ")
    indices = [i for i, x in enumerate(words) if "sr-cit" in x]
    return [application_identifier + "_" + words[i][words[i].find("sr-cit")
            +6:words[i].find("sr-cit")+10] for i in indices]

def main(file):
    f = open(file, "r", encoding="utf8", errors='ignore')
    lines = f.readlines()
    records = {}
    citations = {}
    pattern_en = re.compile("\t20(1[0-9]|20)-\d{2}-\d{2}\ten\t")
    flag_d = False
    flag_c = False
    flag_t = False
    flag_a = False
    flag_am = False
    flag_ams = False
    temp_record = {}
    for line in lines:
        if re.search(pattern_en, line):

            if "\tTITLE\t" in line:
                if flag_c:
                    flag_a = False
                    flag_d = False
                    flag_c = False
                    application_identifier = line.split("EP\t")[1].split("\ten\t")[0].replace("\t","")
                    application_title = line.split("\tTITLE\t")[1][2:]
                    flag_t = True
                    temp_record = {}
                else:
                    application_identifier = line.split("EP\t")[1].split("\ten\t")[0].replace("\t","")
                    application_title = line.split("\tTITLE\t")[1][2:]
                    flag_t = True

            if "\tABSTR\t" in line and flag_t:
                application_abstract = line.split("\tABSTR\t")[1]
                flag_a = True

            if "\tDESCR\t" in line and flag_t:
                application_number = line.split("EP\t")[1].split("\t")[0]
                application_category = line.split("EP\t")[1].split("\t")[1]
                application_date = line.split("EP\t")[1].split("\t")[2]
```

Continuation of the function on next page

```python
            if not flag_a:
                temp_record = {
                    application_identifier : {
                        "application_number": application_number,
                        "application_category": application_category,
                        "application_date": application_date,
                        "title": application_title,
                        "description": line.split("\tDESCR\t")[1][7:]}}
                flag_d = True
                if application_date == "":
                    #print("Skipping entry, missing date: " + application_identifier)
                    continue
            elif flag_a:
                temp_record = {
                    application_identifier : {
                        "application_number": application_number,
                        "application_category": application_category,
                        "application_date": application_date,
                        "title": application_title,
                        "abstract": application_abstract,
                        "description": line.split("\tDESCR\t")[1][7:]}}
                flag_d = True
                if application_date == "":
                    #print("Skipping entry, missing date: " + application_identifier)
                    continue
        if "\tCLAIM\t" in line and flag_d and application_identifier in temp_record:
            flag_c = True
            temp_record[application_identifier]["claims"] = line.split("\tCLAIM\t")[1][7:]
        if "\tAMEND\t" in line and flag_c and application_identifier in temp_record:
            application_amend = line.split("\tAMEND\t")[1]
            temp_record[application_identifier]["amended_claims"] = application_amend
            #flag_am = True # need to reset
        if "\tACSTM\t" in line and application_identifier in temp_record:
            application_amended_statements = line.split("\tACSTM\t")[1]
            temp_record[application_identifier]["amended_claims_statements"] = application_amended_statements
            #flag_ams = True
        if  "\tSRPRT\t" in line and application_identifier in temp_record:
            # if extract_classifications(line):
                # and extract_citationIDs(application_identifier, line):
            temp_record[application_identifier]["citation_ipcr_classification"] = extract_classifications(
                line)
            temp_record[application_identifier]["citation_ids"] = extract_citationIDs(
                application_identifier, line)
            temp_record[application_identifier]["citation_application_number"] =
            from_citationIDs_to_application_number(line)
            application_number_and_category = application_number + application_category
            for citation_id in temp_record[application_identifier]["citation_ids"]:
                #print("evaluate citation id: "+citation_id)
                citations[citation_id] = extract_citation_entry(
                    citation_id,
                    line.split("\tSRPRT\t")[1],
                    application_number_and_category)
        if "\tPDFEP\t" in line and application_identifier in temp_record:
            application_url = line.split("\tPDFEP\t")[1]
            temp_record[application_identifier]["publication_url"] = application_url
        if  application_identifier in temp_record: # and "citation_ids"
            records.update(temp_record)

print(f"File {file} finished")
upload(records, INDEX_APPL, "patent_eu")
upload(citations, INDEX_CIT, "citation_eu")
```

7.3 ElasticSearch

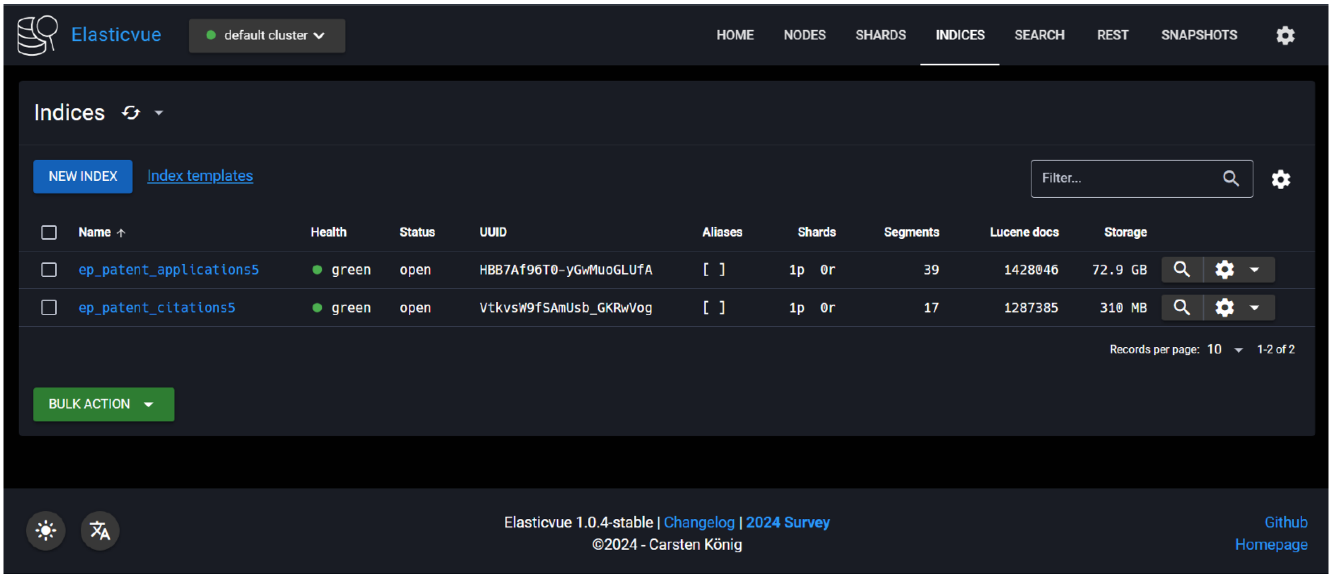


Figure 1: ElasticVue visualization of indexed datasets ep_patent_applications & ep_patent_citations.

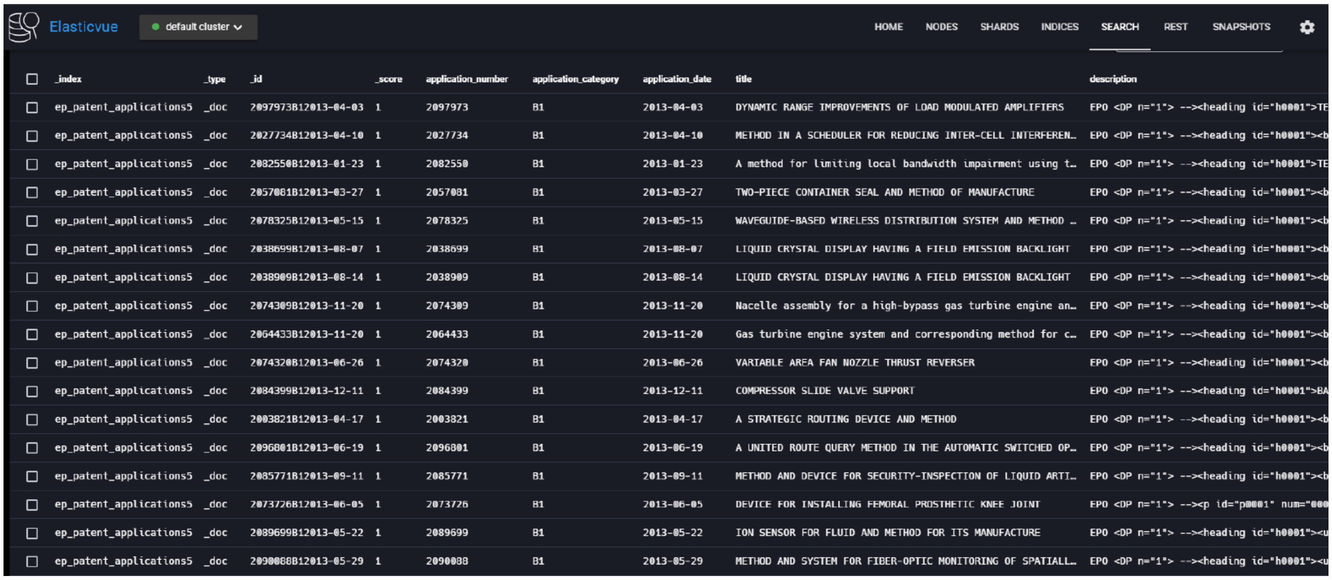


Figure 2: Visualization of ep_patent_applications dataset.

7.4 Q7-Dataset

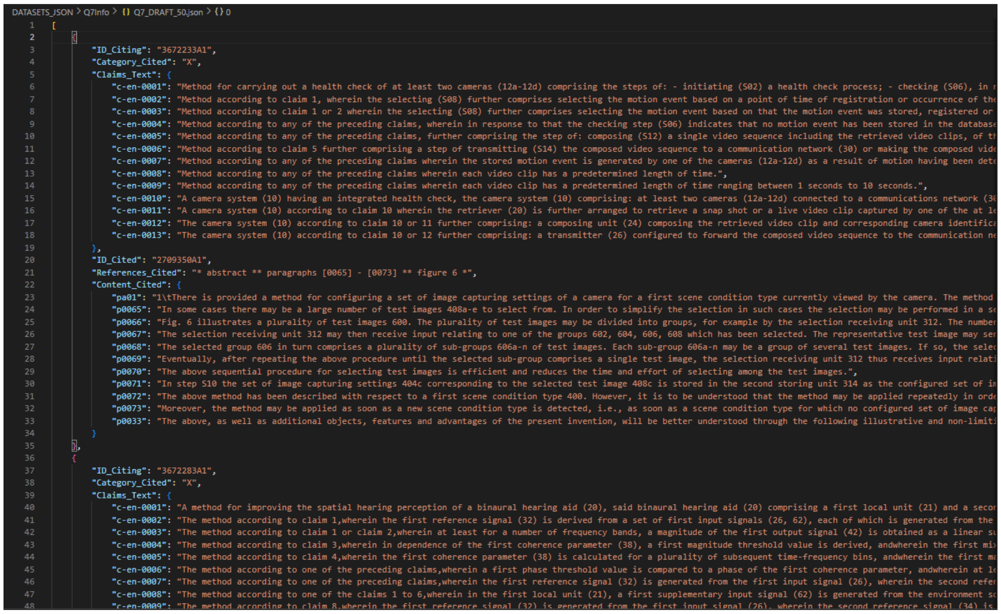


Figure 3: Visualization of Q7's content dataset.

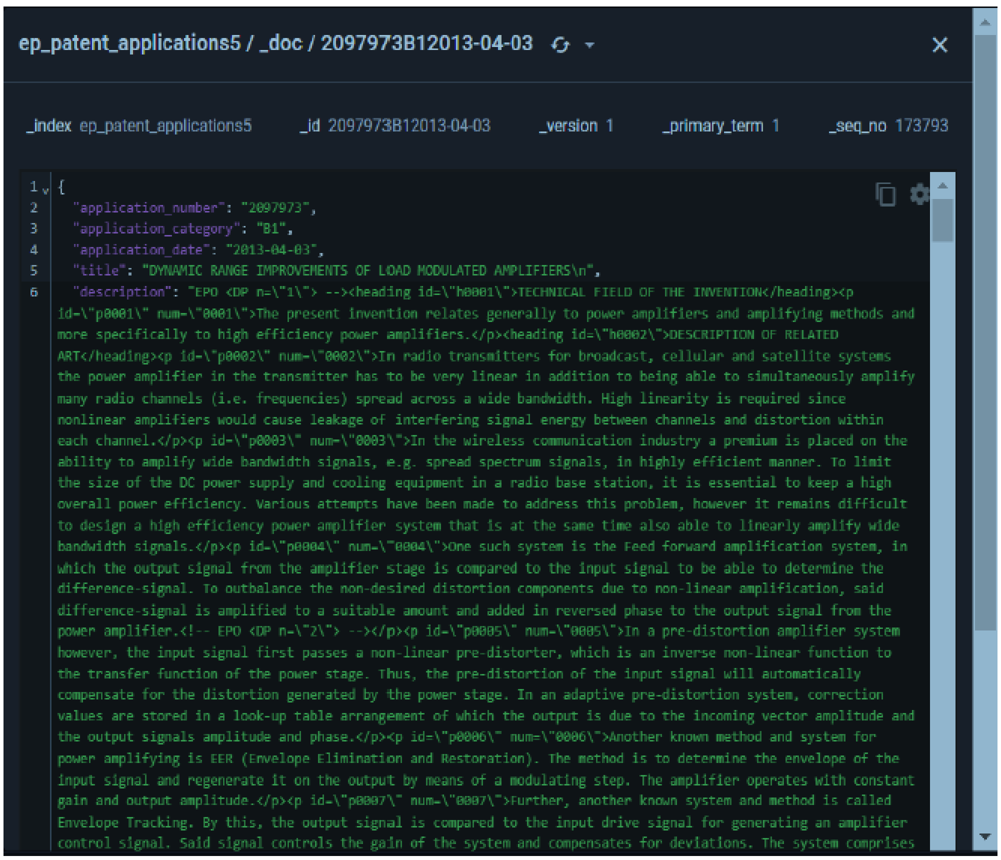


Figure 4: Visualization of raw content information from the dataset ep_patent_applications.

7.5 Q7.py-Script

Some important functions created to extract the paragraphs, claims or figures content.

```
def extract_and_expand_numbers(text):
    pattern = re.compile(r"\[(\d{4})\](?:\s*-\s*paragraph\s*\[(\d{4})\]|\s*-\s*\[(\d{4})\])?")
    matches = pattern.findall(text)
    numbers = []
    for match in matches:
        start = match[0]
        end = match[1] if match[1] else match[2]
        if start and end:
            start_num, end_num = int(start), int(end)
            numbers.extend(range(start_num, end_num + 1))
        elif start:
            numbers.append(int(start))
    return sorted(set(numbers))

def clean_text(html_text):
    clean_text = re.sub(r'<[^>]+>', '', text)

def extract_paragraphs_by_ids(text, paragraph_ids=None):
    extracted_paragraphs = {}
    # Update to match any paragraph ID if None provided
    if paragraph_ids:
        for pid in paragraph_ids:
            pattern = re.compile(r'<p id="p{:04d}" num="\d+">(.*?)</p>'.format(int(pid)), re.DOTALL)
            match = pattern.search(text)
            if match:
                paragraph_text = match.group(1)
                paragraph_text = re.sub(r'<figref idref="[^"]+">(.*?)</figref>', r'\1', paragraph_text)
                paragraph_text = re.sub(r'<[^>]+>', '', paragraph_text)
                extracted_paragraphs[f"p{pid:04d}"] = paragraph_text
    else:
        paragraph_pattern = re.compile(r'<p id="p(\d+)"[^>]*>(.*?)</p>', re.DOTALL)
        matches = paragraph_pattern.findall(text)
        for pid, pcontent in matches:
            # Clean up <figref> tags and other HTML tags from paragraphs
            pcontent_clean = re.sub(r'<figref idref="[^"]+">(.*?)</figref>', r'\1', pcontent)
            pcontent_clean = re.sub(r'<[^>]+>', '', pcontent_clean).strip()
            extracted_paragraphs[f"p{int(pid):04d}"] = pcontent_clean
    return extracted_paragraphs

def split_text_with_complex_pattern(text):
    pattern = re.compile(
    r'\.(?!\s*(Fig|Figs|FIG|FIGS)\.\s*\d+([-,]\s*\d+)*\s+|<\/?\s*(li|ul|p|!--)>)\s*(?=[A-Z])'
    )

    # Split the text using the defined pattern
    lines = pattern.split(text)
    #print("Lines:", lines)
    # Ensure that we filter out any None before processing
    lines = [line for line in lines if line is not None]

    return lines
```

```python
def extract_claims_from_paragraphs(text):
    claim_numbers = []
    # Check if "claims" is mentioned without specific numbers
    if re.search(r'\bclaims?;|\bclaims?$', text, re.IGNORECASE):
        return [0]
    # Pattern to find claims and their ranges, correcting for spaces around dashes
    claims_pattern = re.compile(r'\bclaims?\s*([0-9,-]+)', re.IGNORECASE)
    text = re.sub(r'(\d+)\s*-\s*(\d+)', r'\1-\2', text)  # Remove spaces around dashes in ranges
    text = re.sub(r'\s*-\s*', '-', text)  # Remove spaces around dashes in ranges
    #print(text)
    claims_matches = claims_pattern.findall(text)

    for match in claims_matches:
        claim_list = match.split(',')
        for claim_range in claim_list:
            claim_range = claim_range.strip()
            if '-' in claim_range:
                parts = claim_range.split('-')
                # Ensure both parts are valid before proceeding
                if len(parts) == 2 and parts[0].isdigit() and parts[1].isdigit():
                    start_claim, end_claim = map(int, parts)
                    claim_numbers.extend(range(start_claim, end_claim + 1))
            elif claim_range.isdigit():
                claim_numbers.append(int(claim_range))

    return sorted(set(claim_numbers))

def extract_lines_with_figure_references(text):
    # Remove HTML tags but keep their content, especially for <figref>
    cleaned_text = re.sub(r'<figref idref=\"\w+\">', '', text)

    # Some cleaning
    cleaned_text = cleaned_text.replace('</figref>', '')

    # Lines split correctly
    well_split = split_text_with_complex_pattern(cleaned_text)

    # Keywords to filter
    keywords = ["figure", "figures", "Figure", "Figures", "FIG.", "FIGS.", "Fig.", "Figs."]

    figure_sentences = [line for line in well_split if any(keyword in line for keyword in keywords)]

    lista = []
    for text in figure_sentences:
        # Case 1: Removing <figref> including brackets and extracting the figure number or range directly.
        text = re.sub(r'\[\s*<figref idref="[^"]+">(FIGS?. [0-9a-zA-Z\(\)-]+)</figref>\s*\]', r'\1', text)

        # Case 2: Direct replacement of <figref> tags with their content.
        text = re.sub(r'<figref idref="[^"]+">(FIGS?. [0-9a-zA-Z\(\)-]+)</figref>', r'\1', text)

        # Case 3: Similar to case 2 but specifically targeting the pattern with brackets at the start.
        text = re.sub(r'\[\s*<figref idref="[^"]+">(Fig. [0-9]+)\s*\]?\s*\1</figref>', r'\1', text)

        # Clean remaining HTML tags but keep their content
        text = re.sub(r'<[^>]+>', '', text)
        lista.append(text)

    return lista
```