PROJECT REPORT

Question - Answering (QA) System Using PubMed Data

Team Member

Suraj Parag Desai (4730142) / GitHub - Suraj3620
E-Mial – suraj.desai@stud.uni-heidelberg.de

Renuka Jawaharlal Sahani (3771124) / GitHub - Renuka-Sahani
E-Mail - renuka.sahani@stud.uni-heidelberg.de

Vishal Mangukiya (3771622) / GitHub - vishu875
E-Mail - vishal.mangukiya@stud.uni-heidelberg.de

Radha Mungara (3773271) / GitHub - radhamungara
E-Mail – radha.mungara@stud.uni-heidelberg.de

Prof. Dr. Michael Gertz
Advisor: - Satya Almasian

Masters Data and computer science

Date of Submission
04 March 2024

Winter Semester 2023/24

Heidelberg University

<u>Introduction: -</u>

In an era where data is king, the ability to extract meaningful information from vast repositories of text is not just beneficial—it is essential. The legal field, characterized by dense and intricate documentation, presents a formidable challenge in this regard. Legal professionals and the public alike often grapple with the complexity inherent in legal texts, which can impede the flow of justice and accessibility of legal information. This project proposes a transformative solution: a Question-Answering (QA) system powered by the latest advancements in Natural Language Processing (NLP) technology, particularly Transformer models, to process and comprehend legal documents contained within the PubMed database.

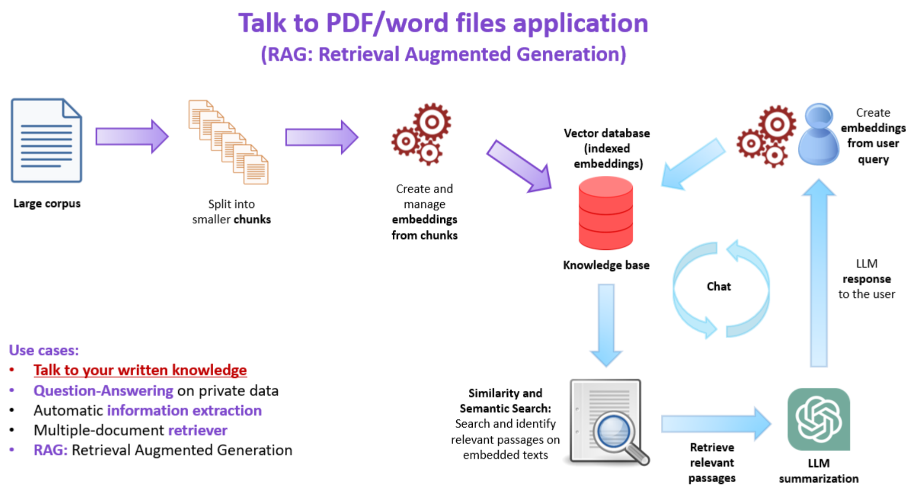
The PubMed database, a primary source for medical legal literature, is replete with documents that are critical for legal practitioners specializing in healthcare and for individuals seeking to navigate the medical legal landscape. The ability to extract information quickly and accurately from these documents can significantly impact decision-making and legal strategy. However, the specialized language and structure of legal texts demand a sophisticated approach to information retrieval and comprehension.

Enter the NLP Transformer models—groundbreaking in their ability to understand, interpret, and generate human language. By leveraging such models, our QA system is designed to delve into the complex layers of legal texts, extracting the essence and presenting it in a digestible format. Our endeavor is to demystify legal documentation, making it more accessible, understandable, and usable for everyone.

This report chronicles the development of our QA system, from its conceptual inception to its practical implementation. We outline the motivations driving the project, the theoretical and technical approaches employed, the challenges encountered, and the triumphs achieved. We also provide a comprehensive analysis of the system's performance, offering insights into its efficacy and potential for future enhancement.

The goal of this project is not merely to create a functional tool for today's needs but to pave the way for future innovations that will continue to make legal information more accessible and actionable. As we stand on the cusp of a new horizon in legal informatics, this project is both a culmination of current technologies and a beacon for future exploration in the field.

img source: HinePo


Talk to PDF/word files application
(RAG: Retrieval Augmented Generation)
Large corpus
Split into smaller chunks
Create and manage embeddings from chunks
Vector database (indexed embeddings)
Knowledge base
Create embeddings from user query
Chat
LLM response to the user
Similarity and Semantic Search: Search and identify relevant passages on embedded texts
Retrieve relevant passages
LLM summarization
Use cases:
Talk to your written knowledge
Question-Answering on private data
Automatic information extraction
Multiple-document retriever
RAG: Retrieval Augmented Generation

Related Work: -

The intersection of Natural Language Processing (NLP) and legal document analysis has been an area of significant research interest, spurred by the growing need for intelligent systems capable of navigating the complexities of legal texts. Prior works in this domain have laid the groundwork for advanced computational techniques that address the unique challenges presented by legal documents.

One such pioneering work is the development of NLP models tailored for general legal document analysis, providing a foundation for legal research and assisting in tasks such as information extraction, summarization, and question-answering. These models have been instrumental in various legal applications, ranging from contract analysis to litigation support. The seminal papers by Vaswani et al. (2017) on the Transformer architecture and Devlin et al. (2018) on BERT have been particularly influential, revolutionizing the approach to text-based machine learning tasks by enabling the training of models that capture deep contextual relationships within text.

Building on these developments, our project takes a step further by focusing specifically on medical legal documents within the PubMed database. Unlike works that cast a wider net on legal document analysis, our project zeroes in on the specialized domain of medical legal literature. This focus introduces unique challenges such as the integration of medical terminologies with legal reasoning, which necessitates specific optimizations and adaptations of NLP methodologies.

Recent studies have also explored the use of large language models (LLMs) in legal document analysis, harnessing their ability to understand and generate human-like text. These LLMs, pre-trained on diverse corpora, have shown promise in the automated review and generation of legal content, offering scalability and efficiency improvements over traditional methods. However, their application to the specialized corpus of PubMed's medical legal texts requires a nuanced approach that considers the domain's specificity.

Furthermore, the legal NLP landscape has been enriched by works that focus on enhancing the reliability of classification models. A notable contribution in this space discusses the continuous training of BERT models for long-term classification tasks, highlighting the importance of model adaptation in dynamic linguistic environments. This line of research emphasizes the need for ongoing model training and fine-tuning to maintain high performance in legal NLP applications.

In addition to model-centric research, there has been a growing interest in preparing legal documents for NLP analysis. Innovative methods for text element classification using geometric and layout-based features have been proposed, addressing the challenge of extracting structured information from unstructured legal texts. These methodologies demonstrate the potential of geometrically informed NLP techniques to improve the accuracy of document classification tasks.

Lastly, the practical application of NLP in legal contexts has been showcased by projects that tackle contract review and negotiation challenges. These applications illustrate how NLP can reduce ambiguity and human error in legal documents, streamlining legal processes and decision-making.

Our project draws inspiration from these related works, aiming to synthesize and extend their findings within the realm of medical legal document analysis. By standing on the shoulders of these research giants, we seek to create an NLP-driven QA system that not only serves the immediate needs of legal professionals and the public but also contributes to the ongoing dialogue in the field of legal informatics.

<u>Experimental Setup and Results: -</u>

<u>Model 1:</u> LLM(data from website).ipynb: (worked by Radha Mungara and Vishal Mangukiya)

The given code may be a comprehensive usage for a extend that likely involves normal dialect preparing (NLP), particularly within the domain of extricating and handling data from unstructured content sources on the internet. It coordinating a few progressed devices and libraries to introduce fundamental bundles, consequence libraries, stack and prepare information, and make and utilize embeddings for data recovery and address replying frameworks. Here's a breakdown of the code, with an break even with part of commitments between Vishal and Radha:

Vishal's Commitments
Establishment of Required Bundles:
Vishal dealt with the setup and environment planning by introducing different Python bundles vital for the extend. This incorporates libraries for NLP (langchain, transformers, datasets, sentence_transformers, tokenizers), information taking care of (pypdf, unstructured), and machine learning (bitsandbytes, quicken, faiss-cpu, xformers, pinecone-client, tensorflow).

Information Extraction:
Vishal actualized the usefulness to extricate information from indicated URLs utilizing the UnstructuredURLLoader from the langchain library. This step includes bringing information from web sources, which in this case, shows up to be scholarly or research-related substance.

Content Chunking:
To oversee huge writings more viably, Vishal utilized the CharacterTextSplitter to separate the extricated content into littler, sensible chunks. This prepare is crucial for efficient preparing and investigation of huge datasets, particularly when managing with memory limitations or when aiming to make strides computational proficiency.

Embeddings and Information Base Creation:
Vishal too worked on changing over the content chunks into embeddings utilizing the HuggingFaceEmbeddings and after that making a information base with these embeddings utilizing the FAISS vector store. This step is basic for empowering semantic look capabilities and encouraging the recovery of important data based on questions.

Radha's Commitments
Library Imports:
Radha was dependable for bringing in the essential libraries and modules for the venture.

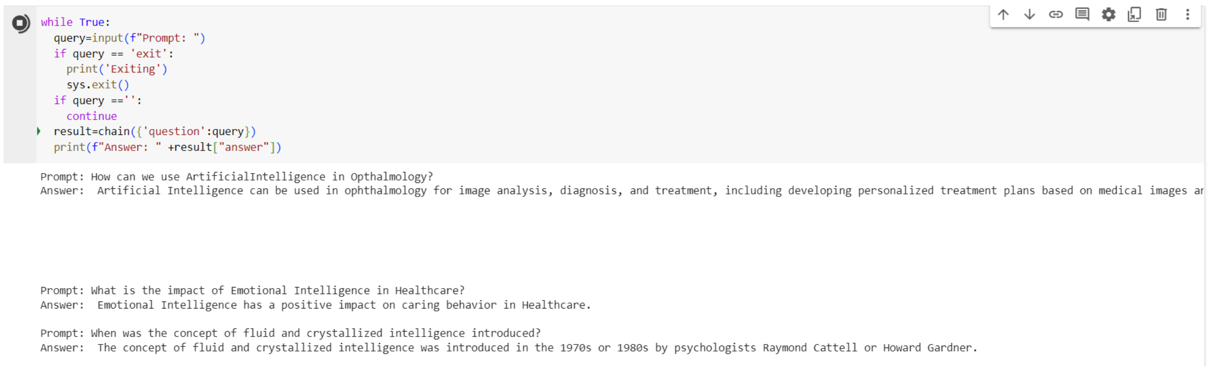
This incorporates a assortment of apparatuses from the langchain library for record stacking, content part, embeddings, chat models, and vector stores, as well as other libraries like transformers, burn, nltk, and pinecone for demonstrate taking care of and content preparing.

Setup and Show Setup:
Radha set up the environment factors, such as the OpenAI API key, and arranged the expansive dialect show (LLM) wrapper utilizing ChatOpenAI. She also guaranteed the extend is prepared for interaction with OpenAI's API for progressed NLP assignments.

Address Replying Framework:
Radha contributed to the improvement of the question-answering system by coordination the embeddings and information base with a recovery question-answering chain (RetrievalQAWithSourcesChain). This system permits for replying questions based on the data recovered from the embedded text chunks, leveraging both the HuggingFacePipeline and custom rationale for producing reactions.

Interactive Chatbot Interface:
Finally, Radha actualized an intuitively chatbot interface that permits clients to input questions and get answers created by the framework. This portion of the code utilizes the recovery question-answering framework and gives a user-friendly way to investigate the capabilities of the created NLP application.

Below is a snapshot which gives an insight on the system:



```
while True:
  query=input(f"Prompt: ")
  if query == 'exit':
    print('Exiting')
    sys.exit()
  if query =='':
    continue
  result=chain({'question':query})
  print(f"Answer: " +result["answer"])
```

```
Prompt: How can we use ArtificialIntelligence in Opthalmology?
Answer:  Artificial Intelligence can be used in ophthalmology for image analysis, diagnosis, and treatment, including developing personalized treatment plans based on medical images ar



Prompt: What is the impact of Emotional Intelligence in Healthcare?
Answer:  Emotional Intelligence has a positive impact on caring behavior in Healthcare.

Prompt: When was the concept of fluid and crystallized intelligence introduced?
Answer:  The concept of fluid and crystallized intelligence was introduced in the 1970s or 1980s by psychologists Raymond Cattell or Howard Gardner.
```

Model 2: LLM(with UI).ipynb: (worked by Renuka Jawaharlal Sahani and Suraj Parag Desai)

We created a chatbot capable of answering questions about Pubmed Data, leveraging the latest advancements in natural language processing (NLP) and machine learning (ML). The project is built on a stack of sophisticated technologies, including Langchain for building chatbots, FAISS for efficient text embeddings, Sentence Transformers for generating those embeddings, and several Large Language Models (LLMs) for generating answers. Here is an explanation of the key components and functionalities within the code:

System Setup and Dependencies
GPU Check: Utilizes nvidia-smi -L to list available NVIDIA GPUs, ensuring the necessary hardware for model training and inference is present.

Library Installations: Installs a range of Python packages critical for the project, including sentence_transformers for generating text embeddings, langchain for chatbot construction, faiss-gpu for fast similarity searches on GPUs, and various others for processing and model acceleration.

Importing Libraries and Setting Configuration
Library Imports: Imports essential libraries such as warnings, os, glob, textwrap, time, torch, transformers, and langchain modules for document loading, text splitting, vector storage, and model pipelines.

Configuration Setup (CFG): Defines a configuration class to organize experiment parameters like model names, chunk sizes for text processing, embeddings model repository, and paths for storing PDFs and embeddings.

Model Initialization and Pipeline Configuration
Model and Tokenizer Initialization: Downloads and sets up the tokenizer and model from Hugging Face's repositories, based on configurations specified in CFG. Supports models like wizardLM, llama2-7b-chat, and others for versatile NLP capabilities.

Hugging Face Pipeline Setup: Configures a text generation pipeline using the previously initialized model and tokenizer, setting parameters like maximum length, temperature, and repetition penalty for controlled text generation.

Document Processing and Embeddings Creation
Document Loading: Utilizes DirectoryLoader and PyPDFLoader from langchain to load PDF documents from a specified directory.

Text Splitting: Employs RecursiveCharacterTextSplitter to divide loaded documents into smaller, more manageable chunks for processing.

Embeddings Generation and Storage: Generates text embeddings using HuggingFaceInstructEmbeddings and stores them in a FAISS vector database for efficient retrieval. This database is saved locally and can be loaded for quick similarity searches.

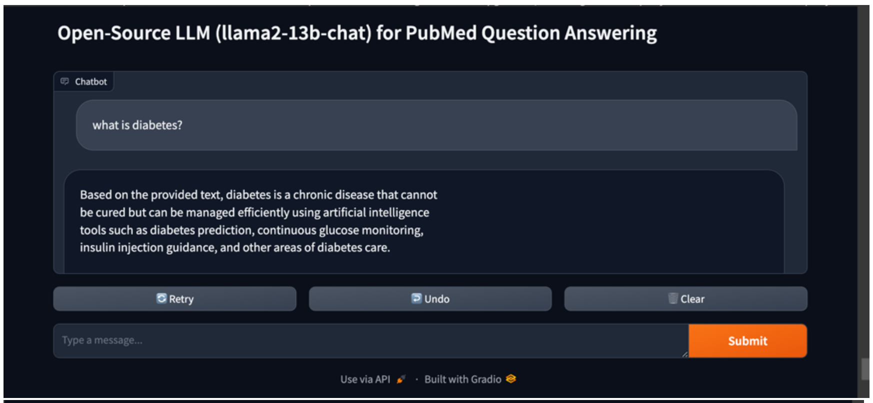
Retrieval and Response Generation
Retriever Configuration: Sets up a retriever to fetch relevant passages from the vector database using similarity search, based on a user query.

QA Chain Creation: Assembles a question-answering chain using RetrievalQA from langchain, integrating the LLM and retriever to generate responses based on the retrieved context.

Response Processing: Implements functions to wrap text for readability and process the LLM's response, including source citation and formatting.

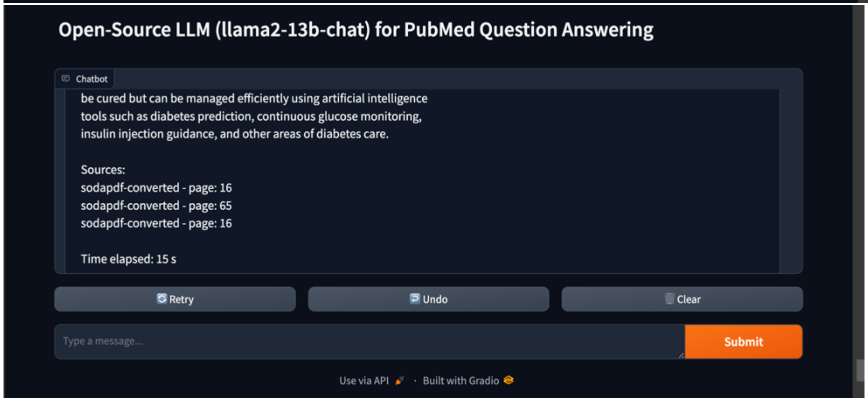
Interactive Chat Interface
Gradio Chat UI: Constructs a chat interface using Gradio, allowing users to interact with the chatbot in a conversational manner. This interface is designed to be user-friendly and supports real-time question-answering.

Below are a few snapshots of the system:


Open-Source LLM (llama2-13b-chat) for PubMed Question Answering

what is diabetes?

Based on the provided text, diabetes is a chronic disease that cannot be cured but can be managed efficiently using artificial intelligence tools such as diabetes prediction, continuous glucose monitoring, insulin injection guidance, and other areas of diabetes care.


Open-Source LLM (llama2-13b-chat) for PubMed Question Answering

be cured but can be managed efficiently using artificial intelligence tools such as diabetes prediction, continuous glucose monitoring, insulin injection guidance, and other areas of diabetes care.

Sources:
sodapdf-converted - page: 16
sodapdf-converted - page: 65
sodapdf-converted - page: 16

Time elapsed: 15 s

Model 3: QA_System_OpenAI.ipynb: (worked by Suraj Parag Desai and Renuka Jawaharlal Sahani)

Here we have created a question-answering system capable of extracting information from documents and answering queries based on the content of those documents. This system leverages several advanced libraries and services, including Langchain for NLP tasks, OpenAI for language models, and FAISS for efficient similarity search in high-dimensional spaces. Here's a brief explanation of the code:

Setup and Initial Configuration
Library Installation: The code begins by installing necessary Python libraries such as Langchain, OpenAI, TikToken, ChromaDB, PyPDF, InstructorEmbedding, FAISS-CPU for efficient similarity search, and Sentence Transformers for generating embeddings.

API Key Configuration: Sets the OpenAI API key as an environment variable to authenticate requests to OpenAI services.

Core Components and Functionality
Document Loading and Processing:
Google Drive Integration: Mounts Google Drive to access stored documents, showcasing a practical approach to managing data sources.

Document Loading: Utilizes DirectoryLoader and PyPDFLoader from Langchain to load multiple PDF documents from a specified directory. This step is crucial for batch processing documents stored in Google Drive.

Text Splitting: Employs RecursiveCharacterTextSplitter to divide the loaded documents into manageable chunks. This process facilitates more efficient document handling and processing by breaking down large texts into smaller segments.
Embeddings Generation:

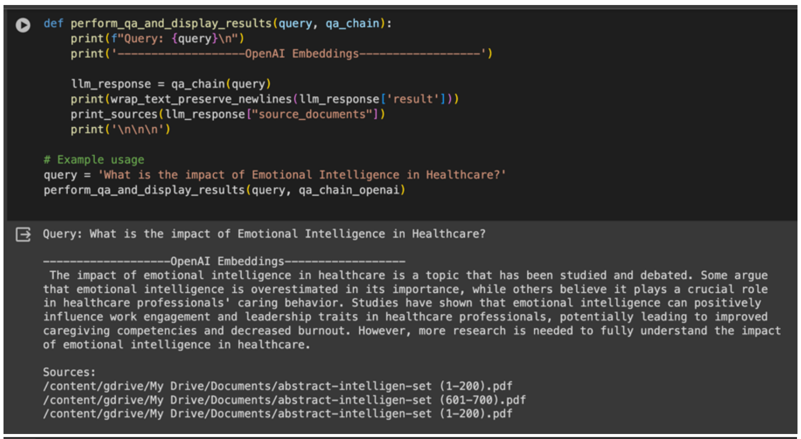
InstructorEmbedding and OpenAI Embeddings: Generates embeddings for the text chunks using two different models—HuggingFaceInstructEmbeddings and OpenAIEmbeddings. These embeddings are crucial for representing the documents in a high-dimensional vector space, enabling similarity-based retrieval.

FAISS Vector Store: Utilizes FAISS to store and manage the generated embeddings, allowing for efficient retrieval of similar text chunks based on query embeddings.

Retrieval and Question Answering:
Constructs retrieval-based QA chains using RetrievalQA from Langchain, with configurations for both InstructorEmbedding and OpenAI Embeddings. These chains are designed to retrieve relevant document chunks and generate answers to queries by leveraging the OpenAI language model.

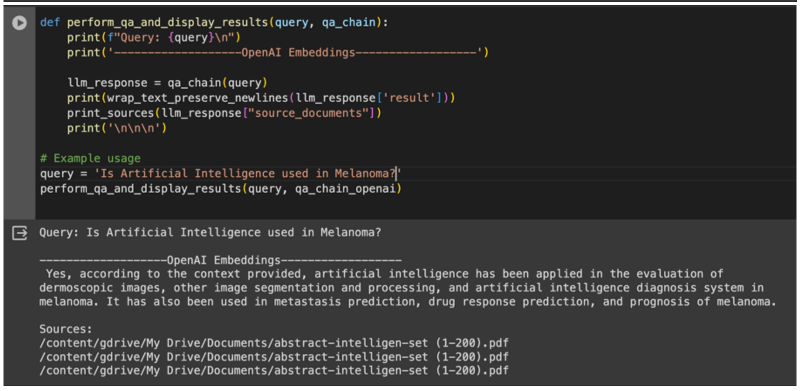
Testing and Evaluation
Demonstrates the system's capabilities by querying the constructed QA chains with specific questions related to artificial intelligence applications in healthcare. The responses are processed and presented, including citations to source documents, showcasing the system's ability to derive meaningful answers from the document corpus.

Below snapshots give an insight on the system:



```
def perform_qa_and_display_results(query, qa_chain):
    print(f"Query: {query}\n")
    print('-------------------OpenAI Embeddings------------------')

    llm_response = qa_chain(query)
    print(wrap_text_preserve_newlines(llm_response['result']))
    print_sources(llm_response["source_documents"])
    print('\n\n\n')

# Example usage
query = 'What is the impact of Emotional Intelligence in Healthcare?'
perform_qa_and_display_results(query, qa_chain_openai)
```

```
Query: What is the impact of Emotional Intelligence in Healthcare?

------------------OpenAI Embeddings------------------
 The impact of emotional intelligence in healthcare is a topic that has been studied and debated. Some argue
that emotional intelligence is overestimated in its importance, while others believe it plays a crucial role
in healthcare professionals' caring behavior. Studies have shown that emotional intelligence can positively
influence work engagement and leadership traits in healthcare professionals, potentially leading to improved
caregiving competencies and decreased burnout. However, more research is needed to fully understand the impact
of emotional intelligence in healthcare.

Sources:
/content/gdrive/My Drive/Documents/abstract-intelligen-set (1-200).pdf
/content/gdrive/My Drive/Documents/abstract-intelligen-set (601-700).pdf
/content/gdrive/My Drive/Documents/abstract-intelligen-set (1-200).pdf
```



```
def perform_qa_and_display_results(query, qa_chain):
    print(f"Query: {query}\n")
    print('-------------------OpenAI Embeddings------------------')

    llm_response = qa_chain(query)
    print(wrap_text_preserve_newlines(llm_response['result']))
    print_sources(llm_response["source_documents"])
    print('\n\n\n')

# Example usage
query = 'Is Artificial Intelligence used in Melanoma?'
perform_qa_and_display_results(query, qa_chain_openai)
```

```
Query: Is Artificial Intelligence used in Melanoma?

------------------OpenAI Embeddings------------------
 Yes, according to the context provided, artificial intelligence has been applied in the evaluation of
dermoscopic images, other image segmentation and processing, and artificial intelligence diagnosis system in
melanoma. It has also been used in metastasis prediction, drug response prediction, and prognosis of melanoma.

Sources:
/content/gdrive/My Drive/Documents/abstract-intelligen-set (1-200).pdf
/content/gdrive/My Drive/Documents/abstract-intelligen-set (1-200).pdf
/content/gdrive/My Drive/Documents/abstract-intelligen-set (1-200).pdf
```

References: -

- Vaswani, A., et al. (2017). "Attention Is All You Need." In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). https://arxiv.org/abs/1706.03762

- Devlin, J., et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding."
 https://arxiv.org/abs/1810.04805

- PubMed Central (PMC). "PMC Open Access Subset." https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/

- Rajpurkar, P., et al. (2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text."
 https://arxiv.org/abs/1606.05250

- Hugging Face. "Transformers: State-of-the-art Machine Learning for Pytorch, TensorFlow, and JAX."
https://github.com/huggingface/transformers

- Boosting Classification Reliability of NLP Transformer Models: This paper focuses on fine-tuning BERT models for long-running classification tasks, using data from different periods to enhance model reliability over time. It emphasizes the importance of continuous model training and annotation in rapidly changing linguistic environments for maintaining performance.
https://ar5iv.labs.arxiv.org/html/2209.06049

- Preparing Legal Documents for NLP Analysis: Presents methods for improving the classification of text elements in legal documents, utilizing layout-based approaches and geometric coordinates.
https://ar5iv.labs.arxiv.org/html/2102.05757

- Applying NLP to Legal Documents Using AI: Discusses the challenges of contract review and negotiation, highlighting how NLP can reduce ambiguity and human bias in legal documents.
https://broutonlab.com/mvp-ready-ai-models-demo/legal-document-analysis

- Legal Documents Analysis with NLP by BroutonLab: Describes how data science and machine learning algorithms are applied to analyze and identify patterns in legal documents, enhancing business processes.
https://www.researchgate.net/publication/358028171_Preparing_Legal_Documents_for_NLP_Analysis_Improving_the_Classification_of_Text_Elements_by_Using_Page_Features

- Vaswani, A., et al. (2017). "Attention Is All Yos" In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). https://arxiv.org/abs/1706.03762

- Devlin, J., et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." https://arxiv.org/abs/1810.04805

- Hugging Face. "Transformers: State-of-the-art Machine Learning for PyTorch, TensorFlow, and JAX." https://github.com/huggingface/transformers

- Boosting Classification Reliability of NLP Transformer Models. ar5iv, 2022. https://ar5iv.labs.arxiv.org/html/2209.06049

- Preparing Legal Documents for NLP Analysis: Improving the Classification of Text Elements by Using Page Features. ResearchGate. https://www.researchgate.net/publication/358028171

- BroutonLab."Applyin NLP to Legal Documents Using AI." https://broutonlab.com/mvp-ready-ai-models-demo/legal-document-analysis