Pharmaceutical assay search with AI

A Natural Language Processing (NLP) approach

Master's thesis in Data Science and AI

ALI ALLADIN

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Pharmaceutical assay search with AI

A Natural Language Processing (NLP) approach

ALI ALLADIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Analysis and probability theory
Chalmers University of Technology
Gothenburg, Sweden 2024

Pharmaceutical assay search with AI
A Natural Language Processing (NLP) approach
ALI ALLADIN

Supervisor: Michael Lawson, AstraZeneca AB
Johan Jonasson, Department of Mathematical Sciences
Examiner: Johan Jonasson, Department of Mathematical Sciences

Master's Thesis 2024
Department of Mathematical Sciences
Division of Analysis and probability theory
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Wind visualization constructed in Matlab showing a surface of constant wind speed along with streamlines of the flow.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Pharmaceutical assay search with AI
A Natural Language Processing (NLP) approach
ALI ALLADIN
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Retrieving historical assay data in pharmaceutical research is often restricted by reliance on specific metadata, overlooking the contextual information in associated protocol documents. This thesis investigates the potential of utilizing these plain English protocol documents alongside Natural Language Processing (NLP) techniques to implement semantic search for assays. A baseline TF-IDF model and the Transformer models BERT, SBERT, and Longformer were used to get embeddings of protocol documents from a corpus of historical protocols. Their performance in retrieving relevant historical protocols was evaluated based on key technical criteria, where the TF-IDF models and BERT using the chunking technique showed the best results. However, limitations in the evaluation scope introduce some uncertainty to the findings, highlighting the need for more rigorous validation. Nevertheless, the conclusions suggest that integrating NLP-driven semantic search systems could reduce the time and manual effort required for assay retrieval, even though the current approach may need further refinement for practical application. These insights are a promising foundation for developing AI-powered search systems used for pharmaceutical texts.

Keywords: Pharmaceutical texts, Assays, Semantic Textual Similarity (STS), Artificial Intelligence (AI), Natural Language Processing (NLP), Large Language Model (LLM), TF-IDF, BERT, SBERT and Longformer

Acknowledgements

First and foremost, I express my heartfelt gratitude to AstraZeneca for giving me the opportunity to conduct this project. My sincere thanks also go to my company supervisor, Mike Lawson, whose guidance and support have been invaluable. I appreciate the freedom and independence given to me during this research, which allowed me to learn and grow.

I would also like to thank my academic supervisor and examiner, Johan Jonasson. Your academic experience helped shape this project, and I am particularly grateful for your accommodating nature, especially given the unconventional timeline of my work.

Many other people, both at AstraZeneca and at Chalmers, have played a vital role in supporting me with my thesis, so thank you to all of you.

Finally, I would like to thank my family, who have been my source of support and strength throughout this thesis and my entire academic journey. To Pappa, Mamma, and Isha: your belief in me and your encouragement has been my greatest motivation, so thank you!

Ali Alladin, Trollhättan, November 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

AI	Artificial Intelligence
NLP	Natural Language Processing
STS	Semantic Textual Similarity
TF-IDF	Term Frequency–Inverse Document Frequency
BERT	Bidirectional Encoder Representations from Transformers
SBERT	Sentence-BERT

Contents

List of Figures

List of Tables

1
Introduction

This chapter provides an overview of the project's background and context within the pharmaceutical domain, outlines the primary aim of the research, presents the key research questions, and discusses the project's limitations.

1.1 Background

In the evolving landscape of pharmaceutical research, the rapid development of new drugs is a measure of innovation and a critical factor in responding to global health challenges. AstraZeneca, a leader in the pharmaceutical industry, has developed a sophisticated drug registration system based on a microservice architecture to manage the complex data associated with chemical compounds. Although this system is flexible and scalable, integrating artificial intelligence (AI) could present a chance to improve user experience and productivity.

Assays define the necessary experiments for drug development and play a crucial role in the drug registration process. The retrieval of historical assays at AstraZeneca is today hindered by a search process that relies on specific metadata associated with each assay, making the procedure time-consuming and inefficient. This delays the initiation of new experiments and influences the overall pace of innovation within the company. Assays are described in more detail in section 2.1.1.

On the other hand, there are protocol documents, which are plain English descriptions of the experiments. However, these are underutilised for searching assays, as the system cannot interpret the free text in these documents. If leveraged, however, this could simplify the assay setup process. Protocols are described in more detail in section 2.1.2.

This thesis proposes developing an AI system capable of parsing natural language in protocol documents to understand their context, extract relevant information, and, based on this, retrieve relevant historical assays. The project aims to reduce the amount of manual effort needed from scientists by suggesting integrating this AI system into the assay registration process, which should improve the effectiveness of experimental setups and speed up the assay submission process.

Addressing these challenges through AI introduces a valuable opportunity for gaining new knowledge in Natural Language Processing (NLP) applied to specialised phar-

maceutical language. The insights and methodologies developed could potentially be used across other sectors, not only within AstraZeneca, that manage complex, text-heavy data, providing a blueprint for broader AI integration across the pharmaceutical industry. Thus, this project is not only about improving current processes but also about exploring new frontiers in the application of AI to enhance scientific research workflows.

1.2 Aim

The primary goal of this thesis is to explore and propose the development of an AI system and, more specifically, explore different NLP models that could be suitable to the pharmaceutical domain to simplify document search. The outcome of the project, apart from the research about the different NLP models tested, should be a Minimum Viable Product (MVP) demonstrating the workflow from protocol document to embeddings and search results, as well as the proposal of implementation of a larger scale AI system to conduct the semantic search as demonstrated in the project. The scientist can feed in a new protocol document to the system, and based on the contents of the protocol document, the system will find other historical protocol documents (and the related assays) most similar to the one fed in.

Since this project will explore the effectiveness of NLP methodologies in a specific domain, it involves the theoretical challenge of understanding linguistic structures unique to pharmaceutical texts and developing methods to handle these structures effectively. Furthermore, metrics and evaluation methodologies will be required to assess the system's accuracy, reliability, and efficiency within the context of pharmaceutical research.

1.3 Research questions

The primary problem this research attempts to solve is the inefficiency of historical assay retrieval due to its reliance on specialised metadata knowledge. Filling out new assays from scratch delays starting new trials and slows down the company's overall rate of innovation. Thus, this project aims to create an NLP model to read protocol documents' natural language and improve searchability for historical assays.

Questions / *Hypotheses*:

- How can utilisation of historical protocol documents improve the efficiency and accuracy of assay retrieval at AstraZeneca?

  An AI system utilising historical pharmaceutical protocol documents and advanced NLP techniques should significantly reduce the time required to retrieve relevant assays, making the process more efficient and accurate.

- What models perform best for semantic search on pharmaceutical documents?

  A more complex pre-trained model, like, e.g. SBERT, should perform slightly better than a basic model, but the other out-of-the-box pre-trained models should also perform decently.

- Should the structural information from the protocol document be included or removed during preprocessing?

  For the models to truly match documents based on semantic similarity, rather than the document template, it would be more reasonable to remove the "irrelevant" parts of the documents, like headings and tags describing what kind of text it is. Otherwise, the model might match documents based on structure and template rather than semantic value.

1.4 Limitations

Data quality and availability: The system's quality will depend on the data's quality and consistency (protocol documents). It is already known that the protocol documents can be inconsistent regarding structure and style. There is also no annotated dataset of protocol documents available, which means that the evaluation of the models will be limited. This could lead to suboptimal results. However, the primary expected outcome of the project is a proof of concept. Hence, an indication of success would be sufficient.

Integration with existing systems: The integration of the NLP model with the company's existing drug registration system, which is built using a microservice architecture, could present technical challenges. Ensuring seamless interaction between the NLP model and the current system components, including databases and user interfaces, would require proper planning and development efforts. Furthermore, as the NLP model will be integrated into the existing system and used by many scientists, its scalability must be considered. It would also need to be ensured that the model can handle large volumes of data and multiple requests without significant degradation in performance.

Proper integration with the existing system is an interesting and important aspect that, however, does not fall under the main objectives of this project. Hence, it will not be addressed in depth in the work conducted.

Usage and maintenance: AstraZeneca scientists and researchers might need training to use the new system effectively, and lack of familiarity with the technology could be an obstacle for the company. The AI system may require ongoing maintenance and updates to remain effective. Lastly, the project will rely on external libraries and tools, meaning any changes or issues with these dependencies could impact the system's future performance.

2
Theory

This chapter provides the theoretical concepts necessary to understand the methods and techniques used later in the project. It begins with an overview of the pharmaceutical domain. Then, the concepts of semantic textual similarity and the different approaches to measuring it, such as TF-IDF and the different transformer models, are introduced. Finally, cosine similarity, a metric for comparing vectors in a vector space, is described.

2.1 Domain

The pharmaceutical domain is a complex field that encompasses various aspects of drug discovery, such as development, testing, and regulatory processes. Various professionals collaborate to explore the efficacy and safety of new drugs, which requires detailed documentation and communication.

One notable characteristic of the pharmaceutical domain is its use of specific terminology and language that differs significantly from everyday language. This specialised domain language includes, e.g. scientific terms, chemical names, acronyms, and standardised phrases that describe various aspects of experiments and compounds. As a result, effectively working with pharmaceutical texts requires an understanding of this distinct linguistic landscape.

2.1.1 Assays in pharmaceutical research

Assays are fundamental to the drug development process, serving as structured experiments designed to measure compounds' biological and chemical activity under specific conditions [1]. In pharmaceutical research, assays are, for example, used to evaluate the effectiveness and safety of new drug candidates. They provide a systematic way of assessing how different compounds interact, making them essential for drug discovery and optimising existing drugs.

They are used across various stages of drug development, and by providing detailed descriptions of experimental setups, conditions, and outcomes, they ensure the consistency and reproducibility of experiments. This is important for validating results across different studies and laboratories, ensuring that drug candidates meet the safety and efficacy standards before proceeding to clinical trials.

While the specific contents of an assay may differ depending on the experiment or the drug development stage, the following components are commonly included:

- **Project assay information:** An overview that includes the assay's purpose, the hypothesis being tested, and the broader research project.
- **Assay description:** Details of the experimental design, including methodologies, instruments and procedures. This ensures that the experiment can be replicated and the data collected is reliable.
- **Target information:** Biological or chemical targets, such as specific proteins, cells, or compounds that the assay tests.
- **Control summary:** A report of the measures used to validate the assay results. These are critical for determining whether the experimental outcomes are due to the tested variables or external factors.

The structured, systematic documentation of the experimental process ensures that the resulting data is reliable and valid.

The storage and retrieval of assay data is crucial for maintaining efficiency in drug development efforts. These are typically stored in databases, which allow for the management and access of the experimental data generated over time. The systems differ in complexity, from simple relational databases to more advanced cloud-based platforms capable of handling large datasets.

However, retrieving historical assay data depends on specific metadata, such as project names, assay types, or biological targets. The reliance on metadata makes it difficult to locate relevant assays, especially when the metadata is not consistently applied or when researchers are unfamiliar with the exact terminology used [2]. This can result in delays in retrieving historical data, which slows down the drug development process, particularly when researchers need to reference past experiments for new trials.

2.1.2 Protocol documents

Protocol documents provide a textual description of the experimental procedures used in drug development. They are written in a structured, accessible format and serve as a guide for conducting experiments. They ensure that the researchers understand the experiment's objectives, methods, materials, expected outcomes, etc. Just like assays, protocol documents are crucial in maintaining consistency and reproducibility across experiments.

The typical structure of a protocol document may include the following elements:

- **Introduction:** A brief overview of the experiment, including the purpose and what it is designed to detect.
- **Method descriptions:** Detailed procedures and techniques that must be

followed during the experiment to ensure reproducibility.
- **Equipment, materials, and reagents:** A comprehensive list of the materials, chemicals and equipment required for the experiment.
- **Expected outcomes:** Hypothesis of the results and potential implications.
- **Calculation of results and acceptance criteria:** Information on how the results will be calculated and the criteria for determining whether the outcomes meet the experiment's objectives.
- **Considerations:** Safety, health, environmental and legal factors that must be considered during the experiment.

While most protocol documents typically follow a structure, there can be variability depending on the author and the specific requirements of the experiment. This variability makes it hard to ensure consistency across experiments, as different documents may follow slightly different formats or emphasise certain elements over others.

Additionally, protocol documents often include not only text but also figures and tables. Tables can contain vital data such as quantities, methods, and conditions that are not always captured in the main body of the text. On the other hand, figures often provide visual representations of experimental setups or results, serving as "supplementary" material but generally not essential for the core experimental details.

2.1.3 Relation between assays and protocol documents

Protocol documents and assays are closely linked in pharmaceutical research. Each assay is typically associated with a protocol document, establishing a 1:1 relationship between the two. Protocol documents contain at least as much information as the assays, often even providing a more detailed account of the experimental procedure. While an assay focuses on structured data such as experimental setups, conditions, and results, the corresponding protocol document offers a broader narrative that includes additional contextual details.

Protocol documents are not utilised when retrieving assay data despite their close relationship. In many systems, assays are primarily accessed using metadata-driven searches, which overlook the information stored in the protocol documents. This could be a valuable resource for improving the accuracy and efficiency of search processes.

2.2 Semantic Textual Similarity

STS refers to the process of determining how closely the meanings of text are related. Unlike simple keyword matching or syntactic similarity, which focuses on the presence of identical words or grammatical structures, semantic similarity is about the deeper contextual meaning behind the text. Two sentences can be semantically similar even if they do not share words as long as they convey the same or a closely related message.

For example, the sentences *"The cat is sleeping on the couch"* and *"A cat is resting on the sofa"* do not share all words in common. However, they express very similar meanings, making them semantically similar.

In the field of NLP, STS plays a vital role in tasks such as information retrieval, machine translation, and text summarisation. It enables systems to better understand human language by identifying relationships between different expressions of the same idea. Focusing on the meanings of words and phrases rather than their surface forms, STS allows for more accurate and nuanced comparisons between texts.

Text is converted into a numerical representation called a vector to measure STS computationally. This process involves embedding words, phrases, or entire sentences into a vector space, where each text is represented as a point in that space [3]. The intuition behind this approach is that texts with similar meanings will have vector representations that are closer together, while dissimilar texts will be farther apart. Calculating the distance between these vectors makes it possible to quantify how similar two pieces of text are. Section 2.5 describes one of these distance metrics.

Research in STS has made significant progress with the rise of deep learning models like transformers, which have improved the ability to capture the nuances of language. Models like BERT and its variants have shown high performance in various STS tasks, achieving state-of-the-art results in domains such as social media, legal texts, and general-purpose search [4]. Despite this progress, applying STS within specialised domains, such as pharmaceuticals, remains a relatively underexplored area [5]. The pharmaceutical domain presents unique challenges due to its highly domain-specific language and the importance of precision in experimental descriptions. Hypothetically, accurate STS measures in this domain could greatly benefit many tasks.

2.3 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical technique used in NLP to represent text data. It assesses the importance of a word within a document relative to its occurrence across the entire corpus of documents [6]. TF-IDF aims to highlight frequent words within a specific document that are not shared across all documents, making it useful for tasks like information retrieval and text mining.

The TF-IDF score for a term t in a document d within a corpus of documents D is calculated as the product of two components:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{2.1}$$

Where:

- **Term Frequency (TF)** is the frequency of term t in document d, often defined as:
$$\text{TF}(t, d) = \frac{\text{Number of occurrences of } t \text{ in } d}{\text{Total number of terms in } d}$$

- **Inverse Document Frequency (IDF)** measures how much information the word provides or whether it is common or rare across all documents. It is defined as:
$$\text{IDF}(t, D) = \log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right)$$
N is the total number of documents in the corpus, and $|\{d \in D : t \in d\}|$ is the number of documents containing the term t. Logarithmic scaling helps reduce the effect of high document frequencies.

Thus, the TF-IDF score increases with the number of occurrences of a word in a document but is offset by the word's frequency in the entire corpus, downweighting terms that are common across many documents.

2.3.1 Vector representation of text

Using TF-IDF, each document can be represented as a vector in a high-dimensional space, where each dimension corresponds to a unique term in the corpus. Hence, for a given document, the value of each dimension (term) in the vector is its TF-IDF score. Specifically, if there are n unique terms in the corpus, each document d is represented as a vector:

$$\mathbf{v_d} = [\text{TF-IDF}(t_1, d, D), \text{TF-IDF}(t_2, d, D), \ldots, \text{TF-IDF}(t_n, d, D)]$$

2.3.2 Limitations

While TF-IDF is used in many tasks, it has several limitations. For example, it treats all words independently, ignoring word order and context. Additionally, TF-IDF struggles with synonyms, considering them distinct terms despite their potential semantic overlap. More advanced techniques like word embeddings are often preferred when deeper semantic understanding is required.

2.4 Transformer models

Transformer models are a significant advancement in machine learning, especially NLP, and have become the foundation for many state-of-the-art models in recent years. Introduced in the paper *"Attention Is All You Need"*, the Transformer architecture provides a more efficient of modelling sequences without relying on recurrent or convolutional structures [7].

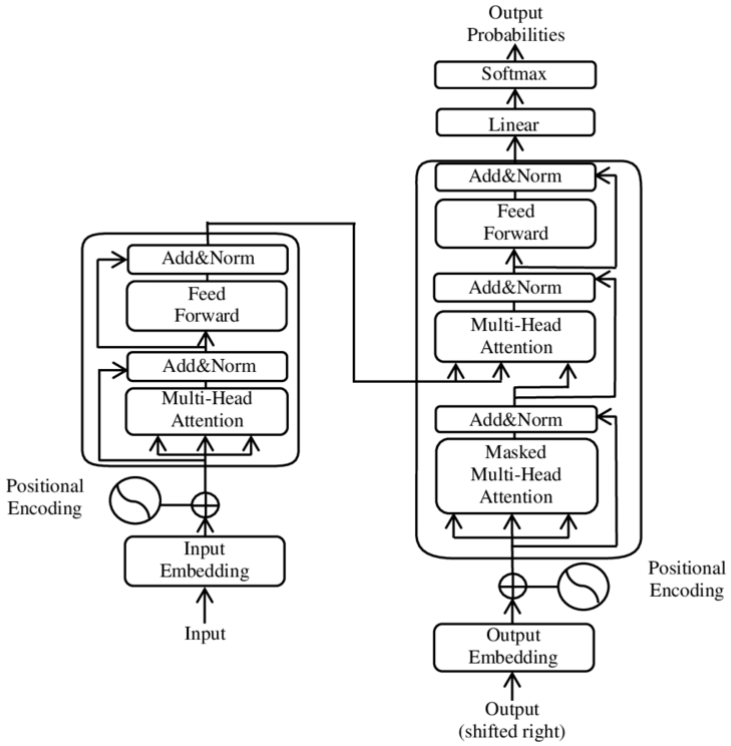


Figure 2.1: The transformer model architecture [8].

Other models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks were more common for processing sequential data before Transformers gained popularity. These models can however struggle with capturing long-range dependencies in sequences and require sequential processing, making them computationally expensive.

The critical feature of Transformer models is the self-attention mechanism. Unlike RNNs and LSTMs, which process sequences one step at a time, Transformers use self-attention to simultaneously process all tokens in a sequence. Using the self-attention mechanism, the models can weigh the importance of each token in the input relative to others. It enables it to capture relationships between words, regardless of their distance in the sequence. This means that a Transformer can understand context more effectively by determining which parts of the input are relevant to each other.

Another advantage of the Transformer architecture is its ability to process all elements of a sequence in parallel. Traditional RNNs require sequential processing, which can be slow for longer sequences. Transformers can handle sequences in parallel, which makes them well-suited for large datasets and complex language tasks.

In addition to the self-attention mechanism, Transformers include multi-head attention, which allows the model to attend to different parts of the input simultaneously, and positional encoding, which provides information about the order of tokens

in the sequence since Transformers do not inherently capture positional information.

Transformers are the foundation for many of today's most successful NLP models, including BERT, which will be discussed in the following section.

2.4.1 BERT

BERT is a Transformer-based model introduced in 2018 [4]. It uses a bidirectional approach to understand the context of words within a sentence and sets a new standard for language representation models. It can capture more nuanced meanings by simultaneously considering both the left and right contexts of each word.

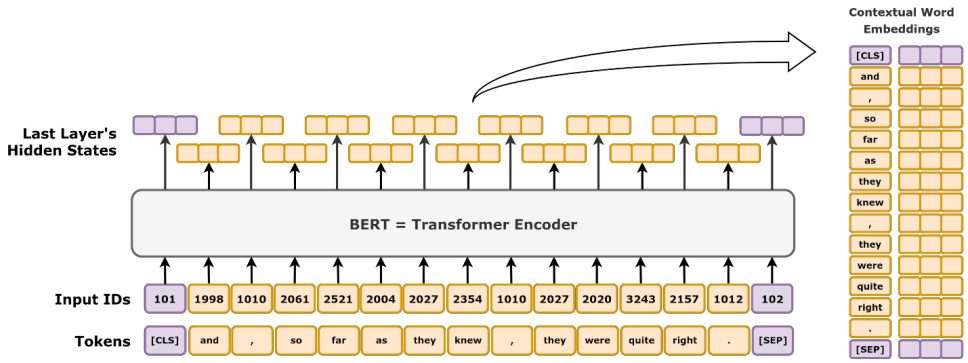


Figure 2.2: High-level schematic diagram of BERT. It takes in a text, tokenises it into a sequence of tokens, adds optional special tokens, and applies a Transformer encoder. The hidden states of the last layer can then be used as contextual word embeddings. [9].

Unlike other models that process text sequentially, BERT processes an entire sentence simultaneously using a self-attention mechanism. This bidirectional nature allows BERT to capture dependencies in language, allowing it to understand each word's context in relation to the others in a sentence.

BERT utilises WordPiece embeddings to represent text. Each input token is transformed into a vector by combining three types of embeddings:

- **Token embeddings:** Each word or subword is represented using embeddings from the WordPiece vocabulary. BERT can handle out-of-vocabulary words by breaking them into subword units, ensuring a better representation of the input text.
- **Position embeddings:** Positional embeddings are added to the input embeddings to capture the order of words within a sentence. This allows BERT to understand the sequence of words.
- **Segment embeddings:** BERT can input paired sentences and use segment embeddings to distinguish between the two sentences. This is for tasks that involve sentence relationships, such as Next Sentence Prediction.

These combined embeddings allow BERT to represent the input text accurately, capturing syntactic structure and semantic meaning.

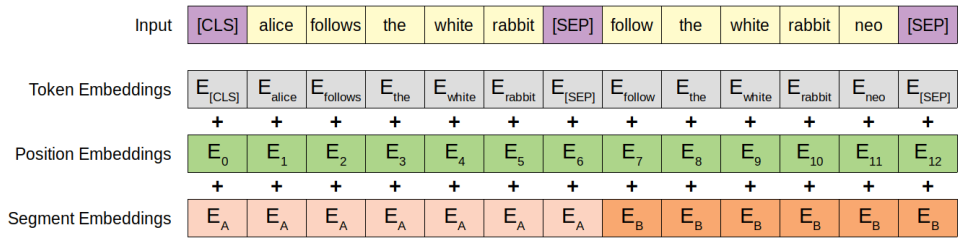


Figure 2.3: The three kinds of embedding used by BERT: token, position, and segment [9].

BERT's training involves two phases: pre-training and fine-tuning. In the pre-training phase, BERT is trained on a large text corpus, including the entire English Wikipedia and the BookCorpus dataset. This phase is unsupervised, meaning the model learns general language patterns without requiring task-specific data. The pre-training itself involves two main tasks:

- **Masked Language Modelling:** In this task, random tokens in the input text are masked, and the model is then trained to predict the masked tokens based on the surrounding context. For example, in the sentence "The cat sat on the [MASK]," BERT learns to predict the word "sofa". This approach allows BERT to capture deep bidirectional representations of language as it learns how words relate to each other in different contexts.
- **Next Sentence Prediction:** The second task trains BERT to comprehend relationships between sentences. Given a pair of sentences, the model predicts whether the second sentence logically follows the first in the original text. For instance, given the sentences "She went to the store" and "She bought some apples", BERT learns to identify the logical sequence. This task helps BERT understand context flow, which is essential for tasks like question answering.

After pre-training, BERT can be fine-tuned on specific downstream NLP tasks using labelled data. Fine-tuning involves adding a small number of task-specific layers to the pre-trained model and updating the weights based on the objectives of the specific task. This allows BERT to adapt its generalised language understanding to perform specific NLP tasks. Due to the comprehensive language knowledge it gained during pre-training, BERT can achieve high accuracy with relatively little task-specific data during fine-tuning.

2.4.2 SBERT

SBERT is a modification of the original BERT model, designed to generate better sentence embeddings efficiently. While BERT sets new benchmarks in various NLP tasks, it is not optimised for direct semantic similarity tasks. BERT requires both sentences to be processed together, making it computationally expensive for large-scale comparisons. For instance, identifying the most similar pair among 10,000

sentences with BERT would require approximately 50 million comparisons, taking around 65 hours on a modern GPU [10].

SBERT addresses this using a Siamese and triplet network architecture, which allows it to generate semantically meaningful sentence embeddings independently for each sentence. This enables quick and efficient similarity computations using measures like cosine similarity. For example, the task of finding the most similar sentence pair in a collection of 10,000 sentences is reduced from 65 hours to about 5 seconds with SBERT, making it more practical for real-world applications.

The key components of SBERT are:

- **Siamese and triplet network structure:** SBERT uses a Siamese network setup, where two BERT models with shared weights deal with each sentence independently to generate embeddings [11]. These embeddings can then be compared using cosine similarity. SBERT also uses a triplet network structure for ranking and retrieval tasks. This allows training the network using three sentences: one sentence is the anchor, the second is positive, and the third is negative. It further enhances the quality of the sentence embeddings by ensuring that semantically similar sentences are mapped closer in the vector space than dissimilar ones [12].
- **Pooling operation:** To produce fixed-size sentence embeddings, SBERT adds a pooling layer to the BERT output. Different pooling strategies are explored, including using the [CLS] token, mean pooling, and max pooling. The default and most commonly used strategy is mean pooling, which averages the token embeddings across the sentence. This method effectively captures the overall semantic meaning of the sentence in a fixed-size vector.
- **Fine-tuning on NLI data:** SBERT is fine-tuned on Natural Language Inference (NLI) datasets such as SNLI and MultiNLI. This training process improves SBERT's performance on tasks like semantic textual similarity by teaching the model to distinguish subtle nuances in sentence meaning.

SBERT has been demonstrated to be effective across various tasks. For example, on several STS tasks, SBERT outperformed models like InferSent and Universal Sentence Encoder, achieving higher Spearman rank correlation scores, which indicates a better understanding of semantic similarity.

SBERT is designed to be computationally efficient. Unlike BERT, which requires pairwise comparisons between sentences, SBERT encodes each sentence independently into a fixed-size vector. This allows for the use of vector-based similarity measures (like cosine similarity), reducing the computational cost significantly. Furthermore, with intelligent batching and leveraging GPU parallel processing, SBERT can substantially increase speed.

While SBERT excels in producing sentence embeddings for tasks where computational efficiency is crucial, it may not consistently outperform traditional BERT

in scenarios where direct pairwise sentence comparisons are more appropriate. For instance, in the Argument Facet Similarity dataset, BERT outperformed SBERT in cross-topic evaluation because BERT's architecture allows for more direct word-by-word comparisons between sentences.

2.4.3 Longformer

Longformer is also a Transformer-based model designed to handle long sequences, addressing a key limitation of traditional Transformer models like BERT: their inability to process long documents. Traditional Transformers use a self-attention mechanism that scales quadratically with the sequence length, making them computationally impractical for long sequences. As a result, models like BERT have a maximum sequence length limit (typically 512 tokens), forcing longer documents to be truncated or split into smaller chunks. This limitation leads to loss of context, particularly in tasks that require understanding the entire document.

Most transformers face a max sequence length problem due to the quadratic scaling for the sequence length. Longformer introduces a novel attention mechanism that scales linearly with the sequence length, allowing it to efficiently process documents with thousands of tokens. The innovation in Longformer combines two types of attention: local windowed and task-motivated global attention.

- **Local windowed attention:** In local attention, each token attends only to a fixed-size window of neighbouring tokens. This approach reduces the computational complexity of the self-attention mechanism by limiting the number of interactions between tokens. Local attention effectively captures context within small regions of the document, allowing the model to focus on relevant nearby information without the need for global context in every layer.
- **Global attention:** In addition to local attention, Longformer has a global attention mechanism. Global attention allows specific important tokens to attend to the entire sequence, ensuring that critical elements in the text can influence and be influenced by all other tokens. These global attention tokens are selected based on the specific task, e.g. unique tokens like [CLS] for classification.

This approach allows Longformer to handle long documents directly, avoiding the need to truncate important content or manually split documents into smaller chunks. As a result, Longformer can retain the full context of long texts, making it useful for document-level NLP tasks where the sequence length far exceeds the limits of traditional Transformer models.

Longformer has been evaluated on various NLP tasks and has demonstrated good performance, particularly on tasks that require understanding long contexts.

2.5 Cosine similarity

Cosine similarity is a metric that measures the similarity between two vectors in a space. It used the cosine of the angle between two vectors, providing a value that ranges from -1 to 1 [13]. A value of 1 means that the vectors point in the same direction (perfect similarity), while 0 indicates that the vectors are orthogonal (no similarity). A value of -1 indicates that the vectors point in precisely opposite directions, though this is less common in most textual similarity tasks.

Given two vectors $\mathbf{A}$ and $\mathbf{B}$, cosine similarity is computed using the formula:

$$\text{cosine\_similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{2.2}$$

Where $\mathbf{A} \cdot \mathbf{B}$ represents the dot product of the vectors, and $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ are the magnitudes (or norms) of the vectors. The dot product measures the extent to which the two vectors are aligned, while the magnitudes normalise this alignment, ensuring that the result is independent of the vector lengths.

Cosine similarity is beneficial in text-based applications because it focuses on the orientation of the vectors rather than their magnitude. This means that two documents with similar content will have a high cosine similarity, regardless of length. This property is especially advantageous when comparing document embeddings, which can vary in length and magnitude depending on how much information they encode.

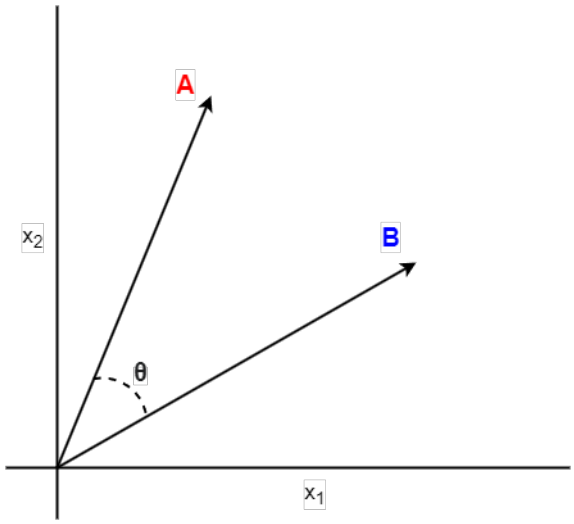


Figure 2.4: Illustration of two vectors in a multi-dimensional space, showing the angle θ between them. The cosine of this angle determines the cosine similarity.

As shown in Figure 2.4, the angle θ between the vectors determines their similarity. A smaller angle corresponds to a higher cosine similarity, indicating that the vectors are more aligned. In the context of document retrieval, using cosine similarity

enables the identification of semantically close documents based on their vector representations in the embedding space.

3
Method

This chapter describes the method for developing and evaluating NLP models to improve the pharmaceutical assay search. It covers the data retrieval, preprocessing, and tokenisation steps applied to protocol documents, followed by the implementation of the models. The model evaluation method using expert feedback is also outlined.

This project was implemented using Python and Jupyter notebooks since they provided an interactive development environment. A variety of standard Python libraries, including `pandas`, `numpy`, and `scikit-learn`, were utilised for data manipulation and processing. Additional specialised libraries were employed for document parsing tasks and model implementation. These will be detailed in the relevant sections. Data

3.1 Data

The data used for this project consists of protocol documents from AstraZeneca's internal systems. These documents are stored as DOC, DOCX, and PDF files and are created by scientists across different departments within the company as part of their experimental procedures. Each protocol document corresponds to a specific assay and contains detailed descriptions of the experimental setup.

The assays are categorised into development, production, and retired. Only retired assays are of interest for this project, as these will not undergo further changes. Currently, over 15,000 retired assays are available, each with a corresponding protocol document. However, only assays from AstraZeneca's Mölndal site were selected for processing to narrow the scope.

One of the main challenges with protocols is their inconsistency in structure and format, as different individuals with varying levels of detail and formatting standards author the protocol documents. The inclusion of non-textual elements such as tables, figures, and metadata within the protocol documents poses further challenges for text extraction. These inconsistencies and the need to process text and tabular data require careful preprocessing to ensure the data is in a suitable format for the language models.

3.1.1 Retrieval

The data extraction process began by loading the relevant information from a JSON file, which served as a dump of the database, into a `pandas` DataFrame for further processing. A few filtering criteria were applied to ensure that only relevant documents were selected. Documents marked as deleted or those not executed at the Mölndal site were excluded. Additionally, only documents with a ”RETIRED” status, which indicates they are no longer subject to changes, were included in the dataset. Once filtered, the DataFrame was reduced to contain only the essential columns for further steps.

The next step involved downloading the protocol documents. A fetcher object was initialised to automate the download process. For each document in the filtered DataFrame, the download URL was constructed using the document's unique ID and version. The fetcher then downloaded each document and saved it in a predefined folder. For documents that failed to download, the corresponding rows of the DataFrame were removed to ensure only valid entries were processed.

Finally, a backup of the resulting DataFrame, which now included the local paths to the downloaded documents, was saved as a Pickle file for future reference.

3.1.2 Parsing

Parsing protocol documents involves extracting and formatting text and tables from various document formats. To handle the requirements of each document format, specialised parsers were created.

The appropriate parser was initialised for each document based on the file type. The parsers identified and processed text, tables, and other structural elements. The parsed content was then updated in the DataFrame, with rows removed if the parsing process failed.

> ”...to ensure accurate reagent mixing. [HEADING] Sample Preparation [NORMAL] The samples used were prepared in a sterile environment to prevent contamination. Each sample was aliquoted into 1.5 mL Eppendorf tubes and stored at -20°C until further analysis. [HEADING] Instrumentation [NORMAL] The experiments were conducted using a spectrophotometer (model XYZ) calibrated according to the manufacturer's instructions. The calibration process involved measuring standard solutions with known concentrations. [HEADING] Data Analysis [NORMAL] Following the completion of the assays, the data was collected and analysed using the software package ABC. Statistical significance was determined using a one-way ANOVA with a p-value threshold set to 0.05...”

Figure 3.1: An example of an extract from a parsed protocol document. The tags present in the extract are highlighted with colour.

Parsing PDF documents presents challenges, primarily due to the format's lack of inherent structural information. Unlike more structured formats such as DOCX, PDFs do not retain clear distinctions between headings, body text, tables, captions, etc. This lack of structure makes it difficult to differentiate between different types of content, leading to potential ambiguities in the parsing process.

The `openparse` library was used to address these challenges. `openparse` is a PDF parsing tool that combines heuristics and layout analysis to deduce structural information from the document [14]. The library processes the PDF file node by node, attempting to identify and extract elements such as text blocks, tables, and figures. For text, `openparse` uses the position and formatting of each block to make informed guesses about whether the text is part of a heading, paragraph, or another structural component. While this approach is not perfect due to the inherent limitations of PDFs, it captures a significant amount of structural information, making the extracted content more valuable.

Tables, in particular, are challenging to parse from PDFs because they are often represented as graphical elements rather than structured data. `openparse` identifies tabular structures based on the relative positioning of text blocks and the presence of lines or boundaries. Once identified, tables are converted into a string representation, preserving as much of the structure as possible. The output is then cleaned and formatted for consistency with the rest of the document's content.

Due to their structure, DOCX documents are significantly more straightforward to parse than PDF files. DOCX files retain explicit formatting information, including clear distinctions between headings, paragraphs, lists, and tables. This structured format allows for precise content extraction, as each element is defined by its corresponding tag within the document's XML structure.

The `python-docx` library was used to parse these documents [15]. This library provides direct access to the document's internal structure, allowing processing of the elements. Text is extracted along with its formatting information, which helps distinguish between headings, body text, and lists. Headings are tagged according to their level, and list items are recognised and retained in their original order.

Tables, explicitly defined in DOCX files, are parsed into a structured format using `pandas` to manage the tabular data, where each table is converted into a string representation. The structured nature of DOCX documents allows for consistency in the output, making it the most straightforward format to work with.

The parsing of DOCX files is generally reliable, and error handling is minimal due to the inherent structure of the format, ensuring that most of the content is correctly extracted.

Lastly, the DOC parser was designed to handle older DOC files. Since direct parsing of DOC files is not as straightforward in Python, the approach involved converting

DOC files to DOCX format using a word processing application in `win32com.client` [16]. Then, the `python-docx` library was utilised to extract the contents following the same workflow as native DOCX files. During the conversion process, the integrity of the document's content was maintained, ensuring that headings, body text, lists, and tables could be accurately identified and formatted.

While this method enabled extracting content from DOC files, the additional conversion step introduced significant processing delays. Each document required manual interaction with a word processing application, which was time-consuming and prone to errors. Because of this, this method proved too slow and inefficient for large-scale document processing. Therefore, the DOC parser was ultimately not used in the final workflow.

A PowerShell script was written to convert all DOC files to DOCX format to speed up the conversion process. The script scanned a specified directory for DOC files and used Microsoft Word's built-in conversion tool to convert each file to DOCX. The script logged each conversion's progress during the process, ensuring errors were handled. After each successful conversion, the script checked whether the DOCX file was created and, if so, deleted the original DOC file. This approach reduced the processing time compared to manual conversion using Python, enabling handling large numbers of DOC files. The DOCX files were then handled just like described earlier.

3.1.3 Pre-processing

The preprocessing of protocol documents was performed in two iterations, each offering two alternatives depending on the model's specific requirements. The goal of preprocessing was to prepare the text data for the model, ensuring that unnecessary noise was removed. Two distinct preprocessing pipelines were designed: one more extensive for TF-IDF and a less aggressive approach for the Transformer models, which have built-in mechanisms to handle certain text complexities.

In the first iteration (called Preprocessing 1), two alternative preprocessing pipelines were developed to accommodate the differences between the TF-IDF and Transformer models:

Alternative 1: Designed for the TF-IDF model, this preprocessing pipeline included extensive text cleaning steps to ensure the model focused on meaningful terms. The steps involved:

1. **Remove URLs:** URLs were removed.
2. **Replace newlines:** Newline characters were replaced with spaces.
3. **Remove special characters:** A predefined set of special characters was stripped out.
4. **Remove repeated characters:** Any sequences of repeated characters (e.g., "!!!" or "aaa") were removed.

5. **Remove stopwords:** Common stopwords were filtered out using NLTK's stopword list.
6. **Stemming:** Words were reduced to their base form using stemming.
7. **Fix spaces:** Extra spaces were normalised, and leading/trailing spaces were trimmed.
8. **Lowercase:** All text was converted to lowercase for uniformity.

Alternative 2: This pipeline was designed for the Transformer models. Since these models are less sensitive to noise, a lighter preprocessing approach was taken:

1. **Remove URLs:** URLs were removed.
2. **Replace newlines:** Newline characters were converted to spaces.
3. **Remove special characters:** Special characters were removed.
4. **Remove repeated characters:** Repeated characters were removed.
5. **Fix spaces:** Extra spaces were normalised, and leading/trailing spaces were trimmed.

After concerns were raised regarding the possibility of the models matching documents based on their templates rather than the actual content, a second iteration of preprocessing (called Preprocessing 2) was performed. This iteration aimed to explore more aggressive preprocessing, stripping out any structural information that could bias the models.

Alternative 1 and Alternative 2: Both alternatives followed the same steps as in the first iteration, however with the following addition:

1. **Remove headings and tags:** Headings and tags (`[HEADING]`, `[NORMAL]`, and `[TABLE]`) were stripped to prevent the model from relying on structural cues.
2. **Remove named entities:** Named entities such as names and dates were removed using Spacy's *EntityRecognizer* [17].

3.1.4 Tokenisation

Tokenisation is a step in the preprocessing pipeline where text is split into smaller units called tokens, which can be individual words, subwords or characters. This enables models to process text by breaking it down into manageable pieces. Tokenisation is performed differently depending on the type of model used.

For TF-IDF, tokenisation is handled implicitly by the `TfidfVectorizer`, which splits the text into tokens based on whitespace and other delimiters. This tokenisation method works well for TF-IDF, where a simple word-based approach is sufficient.

On the other hand, transformer models such as BERT require a more specialised tokenisation process. These models rely on subword tokenisation to handle out-

of-vocabulary words and capture the meaning of rare or complex terms. For this purpose, tokenisers like the `BertTokenizer` from the `transformers` library (Hugging Face) are used. The `BertTokenizer` breaks down text into smaller subword units based on its vocabulary and ensures that even uncommon words are represented meaningfully by combining known subword components.

In addition to splitting the text into tokens, the tokeniser also performs several important tasks for transformer models:

- **Conversion to input IDs:** Each token is converted into a unique input ID based on the tokeniser's vocabulary so the model can process the tokens as numerical data.
- **Adding special tokens:** Special tokens like `[CLS]` (used for classification tasks) and `[SEP]` (used to separate different segments of text) are added to the tokenised input. These tokens help the model understand the input structure.
- **Padding and truncation:** The tokenised text is either padded or truncated to ensure it matches the model's maximum input length (e.g., 512 tokens for BERT). The padding adds `[PAD]` tokens to shorter sequences, while truncation cuts off longer sequences.
- **Returning tensors:** Finally, the tokeniser returns the tokenised input as tensors, ready to be fed into the transformer model for processing.

3.1.5 Truncation and chunking

Transformer models have a maximum token length (e.g., 512 tokens for BERT), which is often much smaller than the length of the average document in the dataset. Truncation is the more straightforward method, where only the first 512 tokens (or the model's maximum allowed number of tokens) are retained, and the rest of the document is discarded. Truncating the document to fit within this token limit may however result in the loss of potentially valuable information from the later parts of the text, especially for lengthy protocol documents.

To address this limitation, an approach called chunking was implemented. Long texts were split into smaller, manageable chunks, each fitting within the model's token limit while maintaining continuity between chunks. This is achieved by allowing overlap between consecutive chunks so that the end of one chunk slightly overlaps with the beginning of the next. This overlap should help the model maintain context across chunks. Additionally, special tokens (such as `[CLS]` for classification and `[SEP]` for segment separation) are added to each chunk, marking the beginning and end of each segment.

There are some concerns about this technique maybe being counterintuitive as taking the average of chunk embeddings may dilute the distinct features of each embedding. However, by using chunking, the idea is that more of the document's content can be preserved, which could improve the model's ability to capture the entire documents,

which is important for tasks like document similarity and retrieval.

3.2 Models

After preprocessing the protocol documents, the next step is to vectorise the entire dataset using different models. While the specifics of each model are covered in the following subsubsections, the general process for embedding documents is consistent across all models.

The process begins with importing the necessary libraries and loading the preprocessed dataset. Next, the specific components, such as the tokeniser and model, are initialised. Depending on the chosen model (e.g., TF-IDF, BERT, SBERT or Longformer), the corresponding tokeniser converts the preprocessed text into tokens.

Once tokenised, embeddings are generated for each document. If the document length exceeds the model's maximum sequence length, chunking is applied to ensure all parts of the document are captured. The resulting embeddings are stored in a DataFrame for in-memory processing or saved to a Pickle file for future use.

The paragraphs below detail how each model is implemented and used to create document embeddings.

3.2.1 Baseline model

The baseline model for this project is based on the TF-IDF method, using the `TfidfVectorizer` from the `scikit-learn` library. The `TfidfVectorizer` was initialised and applied to the preprocessed corpus of documents to create a TF-IDF matrix, representing the importance of words in each document relative to the entire dataset.

The process began by initialising the `TfidfVectorizer`, with default settings for tokenisation, stopword handling, and other preprocessing parameters as required. Once initialised, the vectoriser was fitted on the preprocessed corpus. This step involved transforming each document into a numerical vector, where each entry in the vector corresponds to the TF-IDF score of a specific term within that document.

The resulting TF-IDF matrix was stored for later use in document similarity comparisons and retrieval tasks.

3.2.2 Pre-trained models

To create embeddings for the protocol documents using transformer models BERT, Sentence-BERT (SBERT) and Longformer were used, which have all been described in more detail in section 2.4. While the overall embedding creation process was similar across these models, they differed regarding maximum sequence length and chunking strategies. The process involved initialising each model and its tokeniser,

preparing documents to fit within the model's constraints, generating embeddings, and storing these embeddings for later use.

Model	Max sequence length	Chunking strategy
`bert-base-uncased` [18]	512 tokens	Chunking with 50-token overlap
`all-mpnet-base-v2` [19]	386 tokens	Chunking with 25-token overlap
`longformer-base-4096` [20]	4096 tokens	No chunking, truncation only

Table 3.1: Summary of pre-trained models for embedding creation.

Each model was initialised using the `transformers` library, along with its corresponding tokeniser [21]. The tokenisers converted the preprocessed documents into tokens compatible with each model's requirements.

All documents were both truncated and split into chunks. BERT processed documents up to 512 tokens, splitting longer texts into chunks with a 50-token overlap. SBERT employed chunking with a 25-token overlap for documents exceeding 386 tokens. In the case of Longformer, which supports a maximum sequence length of 4096 tokens, only truncation was used for documents that exceeded this limit. For all models, the embedding of the `[CLS]` token was used to represent the entire document or chunk.

The embedding creation process was computationally intensive, particularly for SBERT with chunking and the Longformer model due to their complex tokenisation and extensive sequence length. Since all computations were performed locally, running the embedding generation in Jupyter notebooks caused the kernel to crash under the heavy workload. The process was, therefore, switched to standard Python scripts, allowing for better resource management. Additionally, intermediate results were saved periodically during the embedding creation. This approach enabled the process to resume from the last checkpoint in case of a crash, avoiding the need to start the entire process again.

Once completed, the embeddings generated by each model were stored in a DataFrame and saved to a Pickle file for future use.

3.2.3 Searching for similar documents

Once embeddings for all documents in the corpus are created, the next step is to search for documents similar to a given query document. The process involves several steps to ensure the search is performed consistently, using cosine similarity as the metric for measuring the similarity between document embeddings.

1. **Preprocess:** The query document is first preprocessed to ensure it follows the same preprocessing pipeline as the documents in the corpus.
2. **Vectorize:** Once the query document is preprocessed, it is tokenised and passed through the same model used for the corpus documents to generate

its embedding. This ensures that the embeddings are comparable in the same vector space.
3. **Distance:** Pairwise cosine similarity is used to calculate the similarity between the query document's embedding and the embeddings of all documents in the corpus.
4. **Sort:** The similarity scores are sorted in descending order, and the top k most similar documents are retrieved, which should represent the most relevant matches in the corpus based on their cosine similarity to the query document's embedding.

3.3 Evaluation

The models used were evaluated to assess their effectiveness in retrieving semantically similar historical protocol documents. Given the domain-specific nature of the documents, domain experts carried out the evaluation process. This section outlines the procedures that was used to conduct the evaluation.

3.3.1 Test documents

A set of representative protocol documents was selected as test queries to serve as the basis for evaluation. These test documents were chosen to cover a range of assays, providing an assessment of the models' ability to retrieve relevant results.

For each of the selected test documents, queries were executed using all model versions. The retrieval process returned the top 30 most similar historical protocol documents for each query. This relatively large number was chosen to account for identical copies of the query document that existed within the historical dataset.

3.3.2 Expert evaluation

The recorded search results for each query-document-model combination were compiled into an Excel file, which was provided to assay scientists for manual evaluation. Because of the time-consuming task, the retrieved documents for a single query document were assessed based on the following criteria:

- **Target:** Whether the retrieved document had the same target a similar target to the query document.
- **Methodology:** Whether the document used a methodology similar to that in the query document.
- **Readout:** Whether the readout techniques in the retrieved document were consistent with those in the query document.
- **Reagent:** Whether the retrieved document uses reagents similar to those mentioned in the query document.
- **Interesting:** Whether the scientist found the result interesting or relevant if they were conducting this query themselves.

For each criterion, the scientists provided a binary response (”Yes” or ”No”). The ”Interesting” criterion was considered a soft variable and it is important to note that the models are not trained or meant to find documents considered similar by scientists but rather expected to match on the technical criteria. This criterion is purely to help understand the results' practical significance.

3.3.3 Aggregation and analysis

After the expert evaluations were completed, the data was aggregated to provide an overview of each model's performance. The proportion of ”Yes” responses for each criterion was calculated for every model-version combination. These scores provided insights into how well each model captured various aspects of semantic similarity, and the ”Interesting” criterion, together with the overlap analysis, could be used to determine a pattern in how the documents are retrieved.

The aggregated results were then analysed to compare the models' performances. This highlighted strengths and weaknesses, and any deviations or unexpected patterns in the results were noted and discussed to learn about potential areas for improvement.

3.4 Practical implementation

The methods and techniques used during this project were primarily for research and prototyping purposes. While they provided valuable insights, there were more efficient and scalable solutions for a full-fledged system.

Given the current methods' limitations in processing speed, storage requirements, and real-time search capabilities, some additional research was conducted to explore suitable approaches for implementing such a system in a larger-scale production setting. The focus was identifying more robust and efficient technologies for data storage and document retrieval, ensuring that integrating the NLP model into a production environment would be feasible and effective.

4

Results

In this chapter, the findings from the different NLP models are presented. Five protocol documents were selected as query documents (11508, 137, 10353, 12888, and 18877). For each query, the top 30 most similar historical documents and corresponding their similarity scores identified by each version of the models were recorded. The detailed search results for each model and query document are compiled and provided in Appendix A.

4.1 Overlap analysis

An overlap analysis was performed to investigate the retrieval similarity across different models and their versions. This analysis was done to understand how consistently different models retrieve identical or highly similar documents to the query document. Since many models demonstrated strong performance in retrieving identical documents, this overlap analysis provided some insights into model behaviour.

Heatmaps displaying the Jaccard similarity between the top 30 retrieved documents for each model version across all five query documents were generated. The Jaccard similarity measures the overlap between two sets as the ratio of the size of their intersection to the size of their union [22]. These heatmaps are included in Appendix B and visualise the extent of overlap between the models.

A set of shared documents was identified for each query document that appeared in the top 30 search results across all models. These shared documents should hypothetically primarily represent the identical historical documents consistently retrieved by all models and versions. The following summarises the common documents for each query:

- **Query 137:** {136, 137, 237, 17711, 17712, 241, 242, 243}
- **Query 10353:** {8408, 10353, 11119}
- **Query 11508:** {11508}
- **Query 12888:** {12710, 12551, 11860, 12757, 12888, 12889}
- **Query 18877:** {18877}

These results show that certain documents are consistently retrieved. For example, Query 137 had eight documents across models, possibly suggesting a solid consensus on document relevance, whereas Query 11508 and Query 18877 had only one

document, the query document itself.

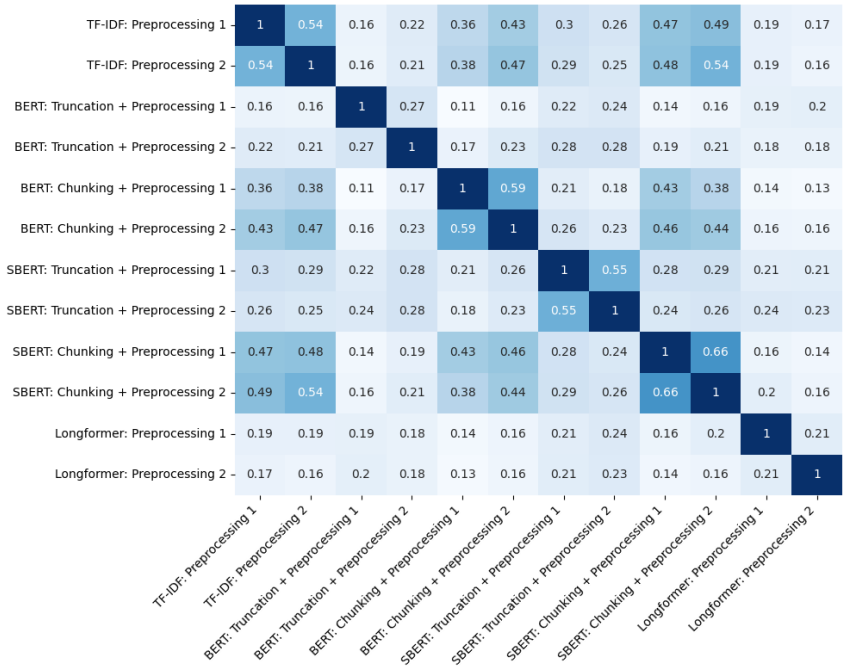


Figure 4.1: Heatmap of average Jaccard similarities across all five queries

The analysis also revealed interesting patterns in the overlap between different models and preprocessing techniques, as seen in Figure 4.1. Notably, a significant overlap was observed between preprocessing 1 and preprocessing 2 within the same model architecture. This is consistent with expectations, as preprocessing 2 is almost identical to preprocessing 1 apart from that it removes, e.g. structural tags and headings) and is designed to further reduce ”noise”.

Another finding is the overlap between BERT chunking and SBERT chunking, which also exhibited relatively high Jaccard similarity scores. This can suggest that despite differences in the underlying model architecture, BERT and SBERT, using the chunking technique, might capture similar semantic information from the protocol documents.

However, outside of these preprocessing and model similarities, there is considerable variation between the results of other models and versions. For instance, models based on truncation show less overlap with those using chunking, likely due to the loss of information caused by truncating long documents. Similarly, Longformer seems to retrieve documents different from the BERT and SBERT models. This spread might indicate that each model prioritises different aspects of the query documents, leading to diverse retrieval results.

4.2 Similarity score distribution

When examining the search results, a significant variation in the similarity scored between different models can be seen. A closer examination revealed patterns in how each model version evaluates document similarity. To explore this further, boxplots of the similarity scores were created for all models (Figure 4.2). These boxplots illustrate a significant spread in the similarity scores across the models, indicating variability in the relevance of the documents retrieved. However, the similarity scores for Longformer models were highly concentrated around specific values, making their boxplots nearly invisible when plotted alongside other models. As a result, the models were separated into different subplots based on their architecture: TF-IDF, BERT, SBERT, and Longformer, as shown in Figure 4.3.

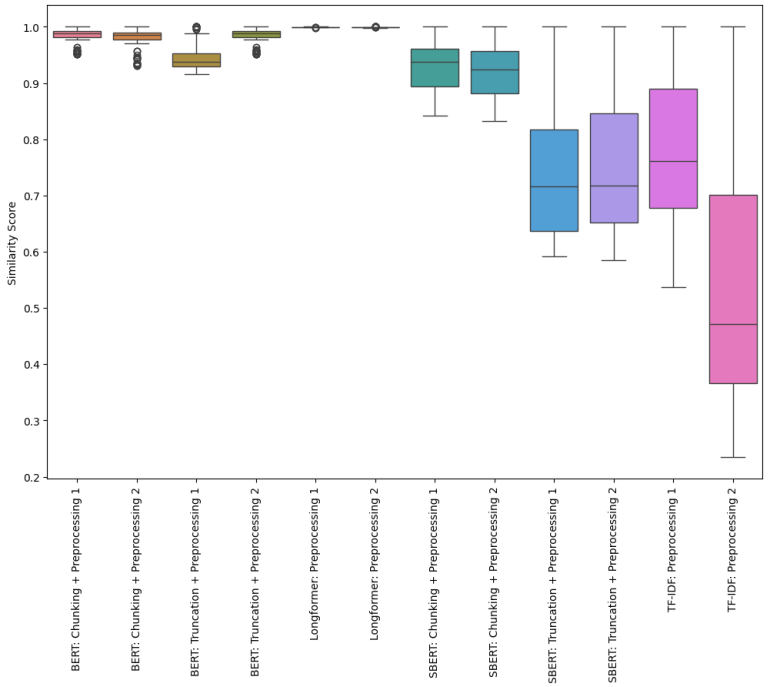
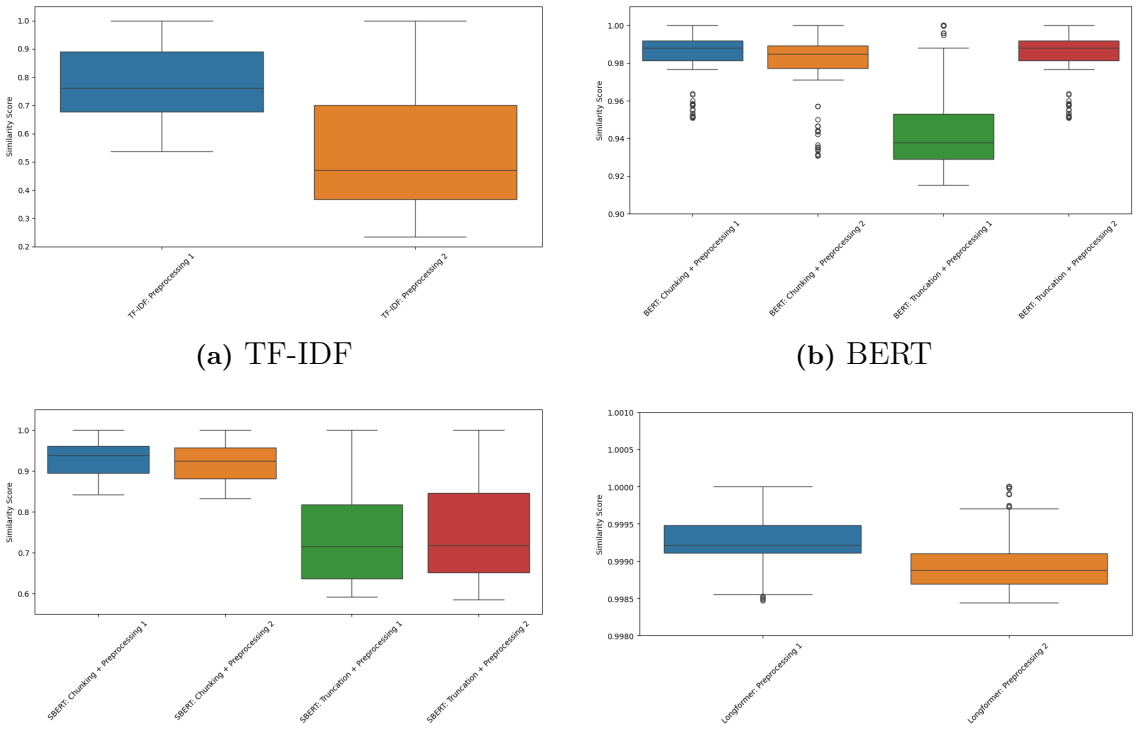


Figure 4.2: Boxplots visualising the distribution of similarity scores for all models.

As can be seen, both Longformer versions exhibited extremely narrow distributions of similarity scores, with values clustered around 0.999–1.0. This should indicate that Longformer retrieves documents that are highly similar to the query, with very little variation in the similarity scores. The tight concentration of scores suggests that most retrieved documents are rated almost identically by the Longformer models, regardless of the specific query. Why this could be the case will be discussed in Chapter 5.

SBERT models, on the other hand, showed more variation depending on the preprocessing technique used. The models that utilised chunking yielded higher similarity scores, with medians around 0.9 and less variation. Truncation on the other hand showed more significant variability in similarity scores, with medians closer to 0.7



(a) TF-IDF **(b)** BERT

(c) SBERT **(d)** Longformer

Figure 4.3: Boxplots visualising the distribution of similarity scores for all models grouped by model type.

and a more comprehensive range of values.

BERT models displayed similar behaviour to SBERT in terms of preprocessing. Chunking resulted in high similarity scores (around 0.98–1.0), indicating highly similar retrieved documents across the queries. The model using truncation and preprocessing 1 demonstrated a wider range of similarity scores, with the median score around 0.94. This variation could reflect the less consistent document retrieval that truncation introduces.

Lastly the baseline TF-IDF models exhibited the most variability in similarity scores among the models analysed. The distributions of scores were broader compared to the transformer models. This wider distribution of scores in TF-IDF could reflect the nature of keyword-based retrieval where the similarity between documents is more sensitive to variations in specific terms rather than the overall semantic content.

Overall, this analysis provides a foundation for understanding each model's behavior. Chapter will discuss the underlying causes and implications of these results.

4.3 Manual evaluation

The manual evaluation of the top 30 search results for query documents provided insights into different models' performance in retrieving relevant protocol documents. The results, summarised in Table 4.1, show the models' accuracy in identifying documents that matched the query document's Target, Methodology, Readout, and Reagent and whether they would be considered "Interesting" by the scientists' perspective. The full evaluation results of document 11508 for each model and version can be found in Appendix C.

	Target	Methodology	Readout	Reagent	"Interesting"
TF-IDF (Preprocessing 1)	100,00%	100,00%	100,00%	100,00%	56,67%
TF-IDF (Preprocessing 2)	100,00%	100,00%	100,00%	100,00%	56,67%
BERT (Preprocessing 1 + Truncation)	63,33%	56,67%	56,67%	60,00%	36,67%
BERT (Preprocessing 2 + Truncation)	60,00%	46,67%	46,67%	56,67%	16,67%
BERT (Preprocessing 1 + Chuncking)	100,00%	100,00%	100,00%	100,00%	63,33%
BERT (Preprocessing 2 + Chuncking)	100,00%	96,67%	100,00%	96,67%	53,33%
SBERT (Preprocessing 1 + Truncation)	90,00%	90,00%	83,33%	80,00%	46,67%
SBERT (Preprocessing 2 + Truncation)	83,33%	83,33%	96,67%	80,00%	43,33%
SBERT (Preprocessing 1 + Chuncking)	83,33%	80,00%	86,67%	80,00%	40,00%
SBERT (Preprocessing 2 + Chuncking)	90,00%	86,67%	93,33%	86,67%	46,67%
Longformer (Preprocessing 1)	86,67%	60,00%	86,67%	60,00%	30,00%
Longformer (Preprocessing 2)	66,67%	53,33%	73,33%	53,33%	43,33%

Table 4.1: Manual evaluation summary with average accuracy for query document 11508.

Many models performed well when returning documents with similar technical criteria. Notably, *TF-IDF (Preprocessing 1)* (see Appendix C.1), *TF-IDF (Preprocessing 2)*, and *BERT (Preprocessing 1 + Chunking)* achieved 100% accuracy across all these criteria, meaning that all 30 retrieved documents had the same or highly similar Target, Methodology, Readout, and Reagent as the query document.

In contrast, BERT models that used truncation preprocessing performed less effectively, with accuracy scores around 50–60% for these criteria (as seen in Appendix C.3). This indicates that truncation likely led to losing crucial contextual information, reducing the models' ability to match documents on these critical aspects.

The "Interesting" criterion is considered a soft variable, as it reflects the judgment of scientists regarding whether a document would be valuable or insightful for their work. For instance, *BERT (Preprocessing 1 + Chunking)* returned 63.33% documents that were deemed "Interesting" by the scientists. This tells that while the model performed well in returning documents with similar technical criteria, only some were considered valuable for further examination. This is because some documents, despite their technical relevance, could have been easily found using traditional metadata-based searches and, therefore, did not offer novel insights.

Compared to the other transformer-based models, *Longformer* performed relatively poorly, with lower accuracy scores across all criteria. *Longformer* models may struggle to capture the most relevant aspects of the protocol documents compared to models like BERT and SBERT.

In general, models that employed chunking, such as *BERT (Preprocessing 1 + Chunking)* and *SBERT (Preprocessing 1 + Chunking)*, seemed to perform better than truncation-based models. However, no significant difference could be seen in the results between preprocessing 1 and preprocessing 2.

4.4 Proposal of integration

Integrating the AI system, which employs the best-performing model, into AstraZeneca's existing microservice architecture would require careful consideration due to differences in the programming languages and technologies. The system should facilitate the retrieval of historical protocol documents through a search interface, providing users with quick access to relevant assays.

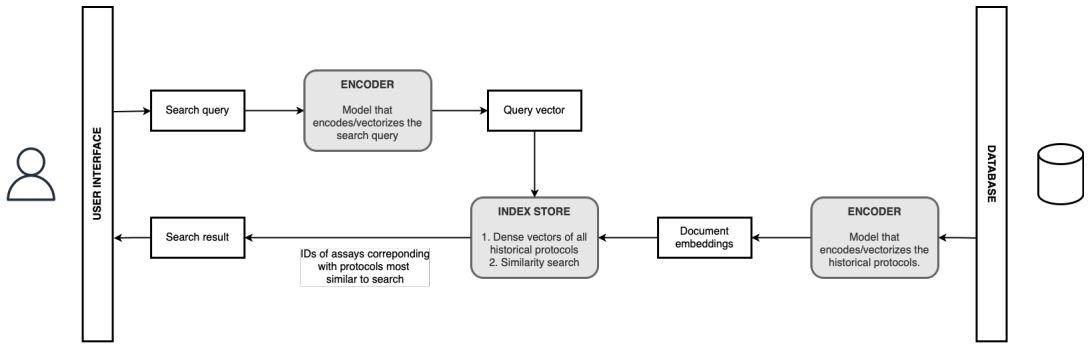


Figure 4.4: A high-level overview of the proposed AI system's architecture.

At a high level, the proposed architecture for the integration consists of key components, including a user interface, a query encoder, an index store, a document encoder and a database, as visualised in Figure B.5. The user interface serves as the entry point for search queries and could be built onto the existing web based AssayCat interface. The query encoder should process the query documents to generate dense vector representations capturing the semantic meaning of the input protocol documents. These vectors would then be compared against precomputed document embeddings stored in the index store (database) to identify the most similar historical protocols. The document encoder would periodically update these embeddings based on the latest protocols available in the database, ensuring up-to-date and efficient retrieval.

One of this integration's main challenges would be bridging the gap between the Python-based AI model and the existing Java-based microservices. A lightweight API gateway is proposed to act as an intermediary layer, exposing the functionalities of the NLP model through simple API endpoints. This API, built using a framework like Flask, would allow Java-based services to interact with the NLP model via HTTP requests, thus decoupling the implementation languages and facilitating seamless communication between components.

The integration also requires an effective method for storing and searching document embeddings. For this purpose, the use of `ElasticSearch` is proposed. ElasticSearch provides an efficient way to store and index vectors and supports advanced similarity search mechanisms (including cosine similarity), crucial for identifying semantically related documents. The embedding generation process, which could be computationally intensive due to the use of transformer models, could be offloaded to AstraZeneca's existing computing platforms to leverage their processing power.

In summary, all this would result in a robust workflow, enabling seamless query submission, similarity search, and results display for scientists.

5
Discussion

This project aimed to explore the application of NLP models to improve the efficiency of assay retrieval at AstraZeneca, addressing the limitations of the current metadata-dependent search system. As mentioned in Section 1, the current system is unable to interpret free text in protocol documents, leading to these not being utilised in locating relevant historical assays. The proposed solution was to develop an AI-driven semantic search system that could understand the content of protocol documents, streamlining the retrieval process and reducing the manual effort required by scientists.

The question of how utilising protocol documents could enhance the efficiency and accuracy of assay retrieval was central to this study. The hypothesis was that advanced NLP techniques could reduce retrieval time by allowing for a more natural and comprehensive search. Second, the study sought to identify which models would perform best for semantic search in the pharmaceutical domain, hypothesising that a complex pre-trained model (e.g., SBERT) would outperform the abseline model. Third, the study considered the impact of structural information in protocol documents on the models' performance, hypothesising that removing elements such as headings and tags would enable models to focus on semantic content rather than structural templates.

5.1 Results and outcomes

By reflecting on the implementation and the results obtained, this part of the discussion will offer insights into the extent to which the research goals were achieved.

5.1.1 Usefulness for scientists

The scientists involved in the project agree that this research is significant, given the challenges of searching among assays. Identifying relevant assays within the vast collection of historical ones is challenging, especially since the existing search methods often fall short. Thus, introducing semantic search methods, which attempt to match documents based on their content, is a step forward in improving how assays are searched and retrieved.

However, there are practical limitations in applying the proposed search method, where an entire protocol document is used as the query. While this approach may

showcase the potential of semantic search technology, it may not fully align with the everyday workflow of scientists. Researchers often do not look for an entire assay that matches another. Instead, they search for more specific information, such as assays with a similar methodology, reagent, or readout. Therefore, this implementation may not directly improve the efficiency of scientists' daily search tasks.

Nevertheless, there are specialised use cases where the current approach, searching for similar documents, could be beneficial. For instance, when an assay encounters problems, scientists may wish to find all other highly similar assays. This could help them identify patterns, troubleshoot issues, or find solutions based on similar assays that have already been conducted. In such scenarios, locating protocols almost identical to those being analysed could be valuable.

5.1.2 Retrieval of identical protocol documents

The initial search results from the overlap analysis revealed a few insights into the models' retrieval performance, particularly regarding their effectiveness in identifying identical documents. One discovery was that many of the top 30 retrieved documents across different models and versions were identical or highly similar to the query documents, as demonstrated in the overlap analysis. For instance, queries such as 137 and 12888 retrieved multiple identical documents across all models, with high consensus. Across most models and versions, there are indications that they perform well in retrieving identical documents to the query document. This indicates that regardless of architecture or preprocessing, most models can recognise and retrieve identical documents.

While this consistency is encouraging in terms of model accuracy, after conversations with scientists, it was revealed that identical documents are often easily retrievable through more straightforward metadata-based searches and, as such, not necessarily as interesting. This could also be seen in the manual evaluation with the soft variable "Interesting". However, there was no mechanism in place to filter out these documents. Hence, when discussing the results, a matching of technical criteria will be considered. However, the documents considered "Interesting" by the scientists reflect a more practical and valuable use case in pharmaceutical assay retrieval.

5.1.3 Performance of baseline model

The TF-IDF model demonstrated the best performance, with 100% accuracy in matching the query document's Target, Methodology, Readout, and Reagent. This indicates the effectiveness of identifying protocol documents with similar technical setups, making it an interesting baseline for comparison with more complex models. TF-IDF probably performed well because of its simplicity and the dataset's very domain-specific language. The simpler approach might capture more relevant features without needing more sophisticated context-aware models.

One observation is that the similarity scores for the TF-IDF model seemed to corre-

late well with the evaluation results compared to other models where the similarity score distribution was concerning. The results suggest that TF-IDF effectively measures similarity meaningfully for this specific query document (11508), even though it uses a relatively simple term-weighting mechanism. Furthermore, the model's performance on the "Interesting" criterion, where 56.67% of the documents were considered interesting by the scientists, shows reasonable behaviour where, after a point, it was not just retrieving identical documents but also documents with practical value.

Additionally, TF-IDF is computationally much more efficient than transformer models. Since TF-IDF operates on a term frequency-based approach, it is much less resource-intensive than transformer models that rely on complex neural networks. This means that TF-IDF could be scaled more easily for large document repositories, offering fast results without requiring significant computational resources, a significant aspect to keep in mind.

5.1.4 Performance of transformer models

As we saw in Section 4.3, the performance of transformer models varied across the different retrieval criteria, with some notable findings. The BERT models using chunking showed good performance, matching the effectiveness of the baseline model. This is probably because chunking, despite the concerns, seems to have allowed the model to process longer sections of the protocol documents so that contextual information was not lost, which likely contributed to better results. This method maintained high accuracy across all critical criteria, with BERT (Preprocessing 1 + Chunking) achieving 100% accuracy in these categories.

BERT performed better than the baseline model in the "Interesting" criterion. With up to 63.33% of the retrieved documents being considered "Interesting" by the scientists. This might suggest that BERT's ability to understand deeper semantic relationships may make it more well-suited for identifying technically relevant documents and those with greater potential value for further investigation by researchers.

Despite initial expectations that SBERT would outperform other models due to its training to minimise cosine similarity between semantically similar documents, its performance was decent but not superior to BERT chunking or even the TF-IDF baseline. Both chunking and truncation achieved similar accuracy in retrieving relevant documents, but they did not stand out. The fact that SBERT's chunking and truncation models gave similar results raises a few questions. A possibility could be that the structure of the documents in this specific dataset did not fully leverage SBERT's pretraining advantages.

The Longformer model, which is designed to handle longer sequences, performed decently but did not outperform either BERT or TF-IDF in terms of accuracy. For instance, its performance in the "Interesting" criterion was noticeably lower, with only about 30–43% of the retrieved documents being considered interesting.

One major downside of Longformer is that it is computationally expensive and requires significantly more resources. In practice, the results may not justify this trade-off, especially since BERT chunking, or even TF-IDF, achieved better results with less computational overhead. There were also concerns about the embeddings produced by the Longformer models. The narrow distribution of similarity scores, as discussed in Section 4.2, suggests that Longformer may not differentiate documents effectively. This could indicate a problem with the embeddings it generates, where the representations of documents are too similar.

There was little difference in performance between Preprocessing 1 and Preprocessing 2 across all transformer models. This result is surprising as Preprocessing 2 involves removing structural information that could have influenced how models understand the documents. One explanation could be that the transformer models can maybe ignore irrelevant structural information and focus on the semantic content. Thus, removing structural elements may not have impacted their ability to retrieve relevant documents, as the models still captured the core experimental content of the protocols.

For BERT, the difference between chunking and truncation was significant, with chunking performing better than truncation, as mentioned earlier. When truncating long protocol documents, valuable contextual information might have been lost, which likely hindered the model's ability to retrieve relevant documents accurately. On the other hand, the chunking approach allowed BERT to process the entire document in smaller, overlapping sections, preserving more critical information. SBERT did not show a significant difference between chunking and truncation, but it is hard to know what could be the reason.

5.2 Limitations of the study

While this study provides valuable insights into using NLP to retrieve historical protocols within the pharmaceutical domain, several limitations must be acknowledged. These limitations have implications for interpreting the results and assessing their contribution to the field.

5.2.1 Unlabeled dataset

One of the primary limitations encountered was the use of an unlabelled dataset. In NLP tasks, labelled data is crucial in model training, fine-tuning, and evaluation. A labelled dataset typically contains annotations that explicitly identify the relevance or similarity of documents, allowing for the use of objective performance metrics, such as precision, recall, and F1-score. However, no pre-existing labelled dataset of pharmaceutical protocol documents was available for this study. As a result, the evaluation of the models had to rely on assessments by domain experts. This reliance on manual evaluation introduces several challenges.

5.2.2 Evaluation method

The evaluation process depended on the expertise of assay scientists, who were asked to review the search results based on predefined criteria (described in section 3.3). While these expert judgements are highly valuable, they inherently contain an element of subjectivity. Different scientists might interpret the criteria differently, leading to variability in the evaluation outcomes. Moreover, this manual assessment approach is time-consuming and limits the number of test queries and retrieved documents that can be evaluated. This constraint further affects the comprehensiveness of the evaluation, as a more extensive set of labelled data would, of course, provide a more robust understanding of the models' performances.

Another limitation was the restricted availability of resources for the evaluation process. Due to the need for expert input, the evaluation was conducted with a small number of assay scientists, whose time and capacity were limited. Consequently, only a limited sample of protocol documents could be used as test queries, and only the top retrieved documents per query were evaluated for each model and version. This small evaluation set may not adequately represent the full spectrum of documents and use cases encountered in real-world applications. For example, the evaluation may not capture edge cases or rare situations where the models perform differently. The limited scope of the evaluation thus affects the generalisation of the findings, as the results obtained from this study might not fully indicate how the models would perform in practice on a broader set of documents.

The results presented in this study are based on an evaluation of a single protocol document, 11508. While the models performed well for this document, particularly achieving high accuracy across key retrieval criteria, these results should be interpreted with caution as they only provide an indication of how the models might perform on a broader set of documents.

An informal evaluation of another protocol document, 12888, suggested that the models may not perform consistently across all protocol documents. For example, in the case of document 12888, most models struggled to retrieve relevant results beyond the identical documents. Longformer models, in particular, returned documents that did not align with the query on any criteria (Target, Methodology, Readout, Reagent, or Interesting). The only model that showed some ability to retrieve relevant, non-identical documents was TF-IDF with Preprocessing 2. This discrepancy between the performance on documents 11508 and 12888 highlights that while some models may perform well on certain protocol documents, they may fail to generalise across all documents in the dataset.

5.2.3 Time and resources

Lastly, the project was constrained by time and resources, which impacted the extent to which experimentation could be performed. While efforts were made to explore a range of models, including fine-tuning transformer-based models, the computational cost of training and evaluating these models meant that only a subset of possible

configurations could be tested. This limitation leaves the possibility that other configurations or models may yield better results. Similarly, the preprocessing steps, such as the decision to include or remove structural information from the protocol documents, were made based on initial experiments. More extensive experimentation with alternative preprocessing strategies might reveal further insights into how best to prepare these documents for semantic search tasks.

Given these limitations, this study's results should be interpreted as preliminary indications. While the models demonstrated potential in retrieving historical protocol documents, the lack of a labelled dataset and the reliance on subjective expert evaluation introduce uncertainty into the findings. The small sample size of test queries and the limited scope of document evaluation further constrain the ability to generalise these results to all use cases within the pharmaceutical domain. Nevertheless, these limitations also highlight important areas for future research.

5.3 Recommendations for future work

Based on the findings while conducting this study, several recommendations can be made for future work to improve the performance and practical utility of the models for document retrieval in the pharmaceutical domain.

One of the challenges identified in this project is the variability in the structure and format of protocol documents. The lack of standardisation complicates the process of parsing and processing these documents, as the models must handle inconsistencies in layout, headings, and formatting. A standardised format for protocol documents would facilitate more effective preprocessing and ensure that the models can identify relevant sections across documents.

Given the extensive length of many protocol documents and the observation that not all sections are equally relevant to scientists, it may be more effective to tailor models to focus on specific parts of the document that are most valuable to scientists or weigh them accordingly. For example, scientists may be primarily interested in sections related to methodologies or readouts. By training models to prioritise these sections, rather than processing the entire document, future systems could improve both the relevance of retrieved documents, computational efficiency and also align more closely with the real-world information needs of scientists at the company.

As mentioned earlier, the results of this study indicated that the transformer models performed decently. However, given these documents' specialised language and context, it is safe to assume that the models do not fully understand and process key terminology and structures. Existing research provides substantial evidence that fine-tuning transformer models on domain-specific corpora can significantly improve their performance. In this case, fine-tuning the models on a large set of pharmaceutical protocol documents could help the models better capture domain-specific language, technical terminology, and the intricacies of experimental protocols. Future work should look into fine-tuning as a step, expecting that doing so could yield

better document retrieval accuracy and relevance results.

Since there is not a labelled dataset with protocol documents that could be used to train an NLP model, one of the techniques *domain-specific pretraining* or *TSDAE* (Transformer-based Denoising AutoEncoder) could enable the models to adapt to the specific language and structure of the protocols. Another alternative is domain-specific pretraining which involves continuing the pretraining process of a model on a domain-specific corpus using tasks like Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) [23]. This process allows the model to adapt to the domain-specific documents' specific terminology and structure. TSDAE, on the other hand, is a technique where a transformer model is trained to reconstruct its input from a noisy version of it [24]. This approach can be used to explore the fine-tuning of language models to produce more robust embeddings, given that there is a large corpus of protocol documents but no labelled data.

In general, a significant limitation of this study was the absence of a labelled dataset. If a labelled dataset would be available some more conventional fine-tuning could be performed. However, apart from fine-tuning, the lack of a labelled dataset restricted the evaluation to manual assessments. Future work should prioritise the creation of a labelled dataset, where protocol documents are annotated with relevance scores based on their similarity to query documents. This would enable more rigorous evaluation across a more comprehensive set of protocol documents. However, if a manual evaluation still needs to be conducted, a longer-term manual evaluation process involving more scientists assessing the relevance of retrieved documents over time would provide deeper insights into model performance in real-world scenarios. A more extensive evaluation would provide a more precise understanding of their effectiveness.

This project aimed to search the assays only based on the protocol documents. However, with the protocol documents, utilising the existing metadata associated with the assays could be an exciting approach. In addition, other machine learning techniques, such as Named Entity Recognition (NER), could also be used to extract and tag key entities from the protocol documents, such as targets, reagents, or read-outs, providing additional data that could improve the search results. Integrating NER with the existing retrieval models could enable a more sophisticated search system that combines semantic similarity with explicit recognition of key entities, leading to more precise and contextually relevant document retrieval.

Bibliography

[1] Enzo Biochem Inc, *What Assays are used for Drug Discovery & Development? - Enzo*, https://www.enzo.com/note/what-assays-are-used-for-drug-discovery-development/, [Accessed 11-09-2024], 2023.

[2] J. Sedlakova, P. Daniore, A. Horn Wintsch, *et al.*, "Challenges and best practices for digital unstructured data enrichment in health research: A systematic narrative review," *PLOS Digital Health*, vol. 2, no. 10, pp. 1–22, Oct. 2023. DOI: 10.1371/journal.pdig.0000347. [Online]. Available: https://doi.org/10.1371/journal.pdig.0000347.

[3] R. Schwaber-Cohen, *Vector Similarity Explained | Pinecone*, https://www.pinecone.io/learn/vector-similarity/, [Accessed 11-09-2024], 2023.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.

[5] J. Gatto, O. Sharif, P. Seegmiller, P. Bohlman, and S. M. Preum, *Text encoders lack knowledge: Leveraging generative llms for domain-specific semantic textual similarity*, 2023. arXiv: 2309.06541 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2309.06541.

[6] "Tf–idf," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2010, pp. 986–987, ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_832. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_832.

[7] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[8] Y. Jia, *The transformer model architecture*, https://commons.wikimedia.org/wiki/File:The-Transformer-model-architecture.png, DOI:10.1088/1742-6596/1314/1/012186, Creative Commons Attribution-Share Alike 3.0 License, https://commons.wikimedia.org/w/index.php?curid=121340680, 2020.

[9] D. V. Godoy, *Bert input embeddings*, https://dvgodoy.github.io/dl-visuals/BERT/, Accessed: Date you accessed the image, 2021. [Online]. Available: https://commons.wikimedia.org/wiki/File:BERT_input_embeddings.png#/media/File:BERT_input_embeddings.png.

[10] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, 2019. arXiv: 1908.10084 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1908.10084.

[11] D. Chicco, "Siamese neural networks: An overview," in *Artificial Neural Networks*, H. Cartwright, Ed. New York, NY: Springer US, 2021, pp. 73–94, ISBN: 978-1-0716-0826-5. DOI: 10.1007/978-1-0716-0826-5_3. [Online]. Available: https://doi.org/10.1007/978-1-0716-0826-5_3.

[12] A. Jana, B. Paudel, M. K. Sarker, M. Ebrahimi, P. Hitzler, and G. Amariucai, "Neural fuzzy extractors: A secure way to use artificial neural networks for biometric user authentication," *Proceedings on Privacy Enhancing Technologies*, vol. 2022, pp. 86–104, Oct. 2022. DOI: 10.56553/popets-2022-0100.

[13] Wikipedia contributors, *Cosine similarity — Wikipedia, the free encyclopedia*, [Online; accessed 27-September-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Cosine_similarity&oldid=1238947356.

[14] S. Filimonov, *GitHub - Filimoa/open-parse: Improved file parsing for LLM's*, https://github.com/Filimoa/open-parse, [Accessed 10-07-2024].

[15] S. Canny, *Python-docx – python-docx 1.1.2 documentation*, https://python-docx.readthedocs.io/en/latest/#user-guide, [Accessed 23-09-2024], 2013.

[16] M. Hammond, *Pywin32 — pypi*, https://pypi.org/project/pywin32/, [Accessed 23-09-2024], 2023.

[17] *EntityRecognizer · spaCy API Documentation*, https://spacy.io/api/entityrecognizer, [Accessed 24-09-2024].

[18] *Google-bert/bert-base-uncased · Hugging Face*, https://huggingface.co/google-bert/bert-base-uncased, [Accessed 03-07-2024].

[19] *Sentence-transformers/all-mpnet-base-v2 · Hugging Face*, https://huggingface.co/sentence-transformers/all-mpnet-base-v2, [Accessed 03-07-2024].

[20] *Allenai/longformer-base-4096 · Hugging Face*, https://huggingface.co/allenai/longformer-base-4096, [Accessed 26-09-2024].

[21] T. Wolf, L. Debut, V. Sanh, *et al.*, "Transformers: State-of-the-Art Natural Language Processing," Oct. 2020. [Online]. Available: https://github.com/huggingface/transformers.

[22] Wikipedia contributors, *Jaccard index — Wikipedia, the free encyclopedia*, [Online; accessed 23-November-2024], 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Jaccard_index&oldid=1254621122.

[23] *MLM — Sentence Transformers documentation*, https://sbert.net/examples/unsupervised_learning/MLM/README.html, [Accessed 04-07-2024].

[24] K. Wang, N. Reimers, and I. Gurevych, *Tsdae: Using transformer-based sequential denoising auto-encoder for unsupervised sentence embedding learning*, 2021. arXiv: 2104.06979 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2104.06979.

A

Appendix A - Search results

A.1 TF-IDF: Preprocessing 1

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	12888	1,000000	18877	1,000000
2	18830	0,916554	136	0,994452	10353	1,000000	12710	0,994716	17234	0,626607
3	18831	0,914864	17712	0,966421	11119	0,998122	12551	0,994347	16900	0,623162
4	12361	0,914085	17711	0,960830	10778	0,997218	11860	0,992827	16934	0,620339
5	12859	0,913604	243	0,897329	19115	0,993999	12757	0,991836	17937	0,604961
6	16766	0,913400	242	0,896195	15506	0,959219	12889	0,894003	18235	0,594951
7	17421	0,912690	237	0,895091	11116	0,937519	13280	0,727764	17995	0,589704
8	16597	0,910889	241	0,894201	13354	0,936150	13447	0,727639	18717	0,581548
9	14372	0,907303	149	0,879710	11148	0,933077	13281	0,727339	18819	0,579694
10	14552	0,906646	148	0,877330	11833	0,923352	13222	0,722975	13358	0,579313
11	16556	0,905169	12851	0,764597	18182	0,890430	13056	0,722797	18820	0,578227
12	16579	0,898800	12850	0,764597	17401	0,890430	22198	0,716210	16618	0,577653
13	10903	0,877005	17539	0,763417	17400	0,890430	12564	0,715349	17993	0,573850
14	16840	0,868747	12674	0,762347	18164	0,888385	22199	0,715294	13994	0,568365
15	16765	0,868486	12675	0,762347	17374	0,888042	15636	0,714630	14597	0,568365
16	15278	0,868024	17541	0,759892	17300	0,888028	12890	0,714124	14344	0,568365
17	19500	0,829629	8456	0,718154	17373	0,887803	11270	0,710460	14313	0,568365
18	19501	0,818377	6903	0,699835	18172	0,884720	12665	0,710289	15609	0,566970
19	11717	0,782407	6892	0,699835	17389	0,884392	20008	0,708390	18239	0,561645
20	11315	0,782407	7652	0,699835	17393	0,884392	10909	0,705948	15182	0,560842
21	11007	0,781105	17512	0,698825	6471	0,825679	14440	0,703019	15183	0,557728
22	11718	0,778699	19005	0,670870	6964	0,825679	14439	0,703019	17549	0,549528
23	11358	0,778699	14123	0,669132	14123	0,800649	15521	0,702964	17259	0,549528
24	18009	0,760440	13787	0,669132	13787	0,800649	15607	0,702069	17295	0,549528
25	10830	0,747803	14204	0,669132	14414	0,800649	15634	0,701715	18866	0,540821
26	12271	0,745581	14414	0,669132	14204	0,800649	12136	0,701664	18867	0,540821
27	9183	0,744462	20008	0,668475	13434	0,797775	13547	0,701630	15389	0,539296
28	6881	0,744146	8541	0,667250	22198	0,797728	12867	0,701630	14689	0,538852
29	7111	0,744146	8539	0,666273	22199	0,796619	10338	0,701540	13049	0,538312
30	17498	0,743587	8540	0,665913	3970	0,794655	13513	0,701270	13418	0,537701

Table A.1: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using TF-IDF with the first iteration of preprocessing.

A.2 TF-IDF: Preprocessing 2

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	10353	1,000000	12888	1,000000	18877	1,000000
2	12361	0,707436	136	0,979072	8408	1,000000	12710	0,965608	17234	0,498045
3	12859	0,706620	17712	0,916607	11119	0,995458	12551	0,964943	16900	0,490598
4	16766	0,706379	17711	0,891338	10778	0,993072	11860	0,962426	16934	0,485755
5	17421	0,706378	243	0,761038	19115	0,982031	12757	0,962216	18235	0,471248
6	18831	0,702670	237	0,757857	15506	0,839091	12889	0,835933	17937	0,470859
7	10903	0,690713	241	0,749798	11116	0,775993	20096	0,347466	17995	0,460031
8	14372	0,660305	242	0,746072	11148	0,764001	11660	0,298420	13358	0,447940
9	16840	0,653334	149	0,705099	13354	0,752119	12127	0,294424	18717	0,447682
10	16765	0,653313	148	0,688297	17401	0,715852	12128	0,293897	18819	0,446913
11	15278	0,653143	17541	0,474683	18182	0,715852	12465	0,292805	18820	0,446157
12	16597	0,650147	17539	0,467506	17400	0,715852	12343	0,283042	16618	0,445452
13	14552	0,643564	12850	0,435436	17300	0,705125	12762	0,282554	17993	0,442645
14	18830	0,623548	12851	0,435436	18164	0,703759	12552	0,281696	14313	0,437972
15	16556	0,606051	17512	0,431119	17374	0,703618	12307	0,281576	14597	0,437972
16	16579	0,599198	12674	0,429728	17373	0,703582	12863	0,281082	13994	0,437972
17	10830	0,584017	12675	0,429728	11833	0,696792	12654	0,279516	14344	0,437972
18	9183	0,581511	8456	0,352399	18172	0,688955	12553	0,279504	15609	0,436940
19	6881	0,581254	7652	0,321344	17393	0,688936	15636	0,255981	18239	0,413526
20	7111	0,581254	6892	0,321344	17389	0,688936	26264	0,242811	12882	0,394491
21	19500	0,563131	6903	0,321344	20252	0,490650	12564	0,242232	12711	0,394491
22	19501	0,545425	15466	0,304258	20251	0,490034	12890	0,241380	13043	0,394458
23	8145	0,507003	15125	0,291117	20253	0,489901	26263	0,239975	13049	0,390688
24	11007	0,488079	15124	0,286237	3031	0,398031	20073	0,239732	13418	0,375512
25	11717	0,488030	19077	0,264727	15418	0,392614	24334	0,239417	13454	0,375512
26	11315	0,488030	19076	0,262120	15419	0,388037	24332	0,238754	17447	0,373204
27	11718	0,487874	19075	0,261370	13000	0,387715	24333	0,234618	17549	0,369827
28	11358	0,487874	7134	0,247697	3889	0,380698	3955	0,234426	17259	0,369827
29	10283	0,480535	7155	0,247697	11199	0,370538	8030	0,234426	17295	0,369827
30	10410	0,480535	7133	0,247697	18475	0,365764	8029	0,234426	19501	0,368747

Table A.2: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using TF-IDF with the second iteration of preprocessing.

A.3 BERT: Truncation + Preprocessing 1

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	11860	1,000000	18877	1,000000
2	16579	0,965460	17712	0,999973	10353	1,000000	12551	1,000000	15183	0,979081
3	21004	0,962400	136	0,999841	11119	0,994725	12710	1,000000	15182	0,966409
4	22472	0,960640	17711	0,999791	11269	0,976602	12888	1,000000	15025	0,959235
5	20578	0,960425	237	0,996088	6471	0,968729	12757	0,977904	15594	0,957728
6	16597	0,960397	242	0,996051	6964	0,968729	12889	0,977904	16788	0,944575
7	18012	0,958937	243	0,995938	6962	0,963867	10390	0,935873	14411	0,943222
8	6467	0,958055	241	0,987977	3887	0,963867	14698	0,933196	18409	0,938316
9	16556	0,957960	6872	0,941859	15475	0,942075	14696	0,933055	12783	0,934328
10	20577	0,956586	13591	0,940115	14972	0,941719	8029	0,929310	19423	0,933499
11	13049	0,953065	20577	0,939663	15503	0,941719	8030	0,929310	18631	0,929326
12	14552	0,952289	20578	0,939440	16934	0,941616	3955	0,929310	14325	0,929300
13	12361	0,951030	17453	0,938570	16618	0,940553	3624	0,928874	12781	0,929267
14	10393	0,948520	14585	0,937671	14571	0,939354	9298	0,927794	15263	0,929210
15	10391	0,948520	6892	0,935215	13008	0,939211	9297	0,927794	20578	0,929059
16	15278	0,948034	6903	0,935215	13007	0,939211	9296	0,927794	18308	0,928393
17	15609	0,947943	7652	0,935215	10778	0,938547	3256	0,923285	12782	0,928190
18	12711	0,947932	12674	0,933462	16766	0,937899	3257	0,923285	19424	0,927544
19	12882	0,947932	12675	0,933462	6470	0,937796	3735	0,923285	21004	0,926925
20	13043	0,947307	12781	0,932432	6804	0,937796	3893	0,923285	14970	0,926911
21	20008	0,947122	12850	0,932330	3891	0,937796	21596	0,922612	20577	0,926064
22	13767	0,947059	12851	0,932330	7156	0,937782	16736	0,921925	15175	0,924676
23	14067	0,947059	17539	0,932142	7154	0,937782	9299	0,921184	18012	0,923263
24	12722	0,946204	12387	0,931733	13543	0,936328	16866	0,920999	16579	0,922658
25	17387	0,945626	14639	0,931667	16900	0,934821	2979	0,919227	22472	0,922388
26	14825	0,945178	14690	0,931667	16765	0,934716	20507	0,917354	12563	0,922030
27	12563	0,944818	17541	0,927850	16840	0,934716	20551	0,916704	11481	0,921506
28	17755	0,944596	9160	0,927845	18164	0,934428	20831	0,915558	12768	0,921383
29	11355	0,944365	9172	0,927845	6881	0,934232	23814	0,915524	11482	0,921293
30	13141	0,944101	9209	0,927845	7111	0,934232	14520	0,915321	8099	0,921107

Table A.3: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *bert-base-uncased* with truncation of protocol document and the first iteration of preprocessing.

A.4 BERT: Truncation + Preprocessing 2

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	17712	1,000000	8408	1,000000	11860	1,000000	18877	1,000000
2	12361	0,994987	137	0,999952	10353	1,000000	12551	0,996933	15183	0,990464
3	12859	0,994560	243	0,996030	11119	0,998798	12710	0,996618	15182	0,990366
4	14552	0,994306	242	0,996009	3892	0,998435	12888	0,996611	14411	0,990322
5	15278	0,994298	237	0,991441	3904	0,997616	12757	0,996516	14325	0,989835
6	16597	0,993986	241	0,990356	10778	0,996439	12889	0,995969	15594	0,989764
7	18012	0,993830	17711	0,988091	6962	0,996138	17679	0,987166	16788	0,989649
8	2972	0,993811	136	0,982661	3887	0,996135	11700	0,987155	18409	0,989254
9	17421	0,993638	12850	0,963711	6471	0,995774	12127	0,987069	15025	0,989254
10	21004	0,993427	12851	0,963353	6964	0,992765	14505	0,986215	14429	0,989238
11	19501	0,993061	12674	0,960075	13043	0,992344	12128	0,985867	18221	0,989169
12	22472	0,992692	12675	0,958426	12711	0,992286	16866	0,985668	12767	0,988335
13	18830	0,992560	8456	0,958204	12882	0,991770	3970	0,984673	14346	0,988202
14	19454	0,992560	6892	0,958178	14554	0,991570	14504	0,982603	14390	0,988132
15	18831	0,991802	6903	0,957874	18276	0,991570	3478	0,980845	21958	0,987917
16	16579	0,991260	7652	0,957725	11833	0,991570	3969	0,980589	18229	0,987832
17	16766	0,991260	148	0,957323	3909	0,991281	14698	0,980047	15609	0,987796
18	20008	0,990893	9210	0,955398	13354	0,990512	13543	0,979272	13076	0,987730
19	20578	0,990861	149	0,955233	19233	0,990512	14696	0,979215	12956	0,987636
20	9308	0,990836	13591	0,954962	19459	0,990333	14348	0,978954	17961	0,987627
21	20577	0,990470	12767	0,953302	18235	0,986808	11276	0,978954	18012	0,987229
22	14372	0,989965	18980	0,953302	13546	0,986087	3470	0,978584	13814	0,985304
23	3431	0,989965	22199	0,952251	19420	0,986018	12048	0,978279	17661	0,985276
24	13141	0,989710	19301	0,951830	18901	0,985957	12711	0,978148	19205	0,985068
25	13352	0,989710	14585	0,951830	22471	0,985905	12882	0,977468	17060	0,984372
26	13745	0,989383	10395	0,951830	13519	0,985837	12465	0,977468	21004	0,983916
27	16893	0,988910	9160	0,951410	19115	0,985835	10403	0,977140	18980	0,982903
28	16583	0,988666	9172	0,951244	17995	0,985735	12136	0,977057	12955	0,982903
29	16892	0,988478	9209	0,951032	16981	0,985637	11877	0,976962	14053	0,982766
30	18229	0,988478	17661	0,950865	3949	0,985610	12181	0,976674	13049	0,982369

Table A.4: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *bert-base-uncased* with truncation of protocol document and the second iteration of preprocessing.

A.5 BERT: Chunking + Preprocessing 1

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	12888	1,000000	18877	1,000000
2	16766	0,994987	136	0,999952	10353	1,000000	12889	0,996933	25850	0,990464
3	10903	0,994560	17711	0,996030	11119	0,998798	11860	0,996618	17234	0,990366
4	12859	0,994306	17712	0,996009	10778	0,998435	12710	0,996611	16900	0,990322
5	12361	0,994298	237	0,991441	19115	0,997616	12757	0,996516	19018	0,989835
6	14552	0,993986	243	0,990356	11116	0,996439	12551	0,995969	19222	0,989764
7	17421	0,993830	242	0,988091	13354	0,996138	11158	0,987166	19317	0,989649
8	14372	0,993811	241	0,982661	15506	0,996135	11541	0,987155	18866	0,989254
9	18831	0,993638	149	0,963711	11148	0,995774	11159	0,987069	18867	0,989254
10	16597	0,993427	148	0,963353	11833	0,992765	13926	0,986215	16934	0,989238
11	8145	0,993061	9308	0,960075	17373	0,992344	11104	0,985867	19309	0,989169
12	18830	0,992692	21002	0,958426	17374	0,992286	11105	0,985668	20461	0,988335
13	10283	0,992560	21003	0,958204	18164	0,991770	11782	0,984673	21995	0,988202
14	10410	0,992560	22471	0,958178	17400	0,991570	20031	0,982603	21871	0,988132
15	15278	0,991802	20942	0,957874	17401	0,991570	13770	0,980845	22201	0,987917
16	6881	0,991260	24550	0,957725	18182	0,991570	19837	0,980589	20543	0,987832
17	7111	0,991260	23565	0,957323	17300	0,991281	19064	0,980047	21861	0,987796
18	18009	0,990893	24025	0,955398	17389	0,990512	19420	0,979272	19017	0,987730
19	10830	0,990861	24045	0,955233	17393	0,990512	15636	0,979215	21994	0,987636
20	9183	0,990836	21004	0,954962	18172	0,990333	19233	0,978954	20601	0,987627
21	11007	0,990470	13767	0,953302	18229	0,986808	19459	0,978954	20452	0,987229
22	11358	0,989965	14067	0,953302	18567	0,986087	12665	0,978584	18235	0,985304
23	11718	0,989965	12781	0,952251	20253	0,986018	17403	0,978279	18239	0,985276
24	11315	0,989710	9160	0,951830	20252	0,985957	11270	0,978148	13043	0,985068
25	11717	0,989710	9172	0,951830	20251	0,985905	14972	0,977468	17937	0,984372
26	12182	0,989383	9209	0,951830	18277	0,985837	15503	0,977468	13049	0,983916
27	11858	0,988910	23823	0,951410	18276	0,985835	12127	0,977140	12711	0,982903
28	12271	0,988666	8456	0,951244	19489	0,985735	12387	0,977057	12882	0,982903
29	17498	0,988478	19724	0,951032	18956	0,985637	17833	0,976962	19983	0,982766
30	12749	0,988478	19104	0,950865	17743	0,985610	12136	0,976674	17743	0,982369

Table A.5: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *bert-base-uncased* with chunking of protocol document and the first iteration of preprocessing.

A.6 BERT: Chunking + Preprocessing 2

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	17712	1,000000	8408	1,000000	12888	1,000000	18877	1,000000
2	16597	0,993628	137	1,000000	10353	1,000000	12710	0,998766	17234	0,987101
3	10903	0,992609	17711	0,996498	11119	0,999221	11860	0,998614	16900	0,985787
4	12361	0,991734	136	0,996498	10778	0,998866	12551	0,998484	19018	0,985535
5	12859	0,991436	243	0,988615	19115	0,998121	12889	0,998226	21871	0,985383
6	18830	0,991406	242	0,988615	11116	0,993649	12757	0,997970	21861	0,985351
7	14372	0,991268	237	0,988154	11148	0,992931	13926	0,982208	20543	0,985316
8	16766	0,990995	241	0,971045	15506	0,992764	11541	0,982175	16934	0,985235
9	8145	0,990455	6892	0,957132	13354	0,992562	11159	0,981766	19309	0,985229
10	17421	0,989953	6903	0,957132	18164	0,989671	11782	0,981660	19017	0,984972
11	10283	0,989779	7652	0,957132	17373	0,989439	11158	0,980885	21994	0,984588
12	10410	0,989779	8456	0,950211	17374	0,989341	11105	0,979978	18866	0,984419
13	10830	0,989778	9160	0,946543	17300	0,989203	11104	0,979536	18867	0,984419
14	14552	0,989584	9172	0,946543	17400	0,988604	15636	0,977604	20452	0,984365
15	16840	0,989321	9209	0,946543	17401	0,988604	12465	0,977603	21995	0,984319
16	16765	0,989316	12850	0,944319	18182	0,988604	12127	0,977019	19222	0,984258
17	16579	0,989197	12851	0,944319	20251	0,987382	12128	0,976540	20461	0,984092
18	15278	0,989013	12674	0,943611	20252	0,987376	14786	0,975710	22201	0,984026
19	6881	0,988750	12675	0,943611	20253	0,987329	14787	0,975710	25850	0,983884
20	7111	0,988750	24550	0,942220	17389	0,987186	12867	0,975573	19317	0,983257
21	9183	0,988745	19223	0,936652	17393	0,987186	13547	0,975573	19983	0,981281
22	16556	0,988628	24045	0,935379	18172	0,987049	11270	0,975545	18468	0,979838
23	11315	0,988128	26181	0,935319	11833	0,986772	11660	0,975192	18235	0,979683
24	11717	0,988128	24025	0,935063	18572	0,981972	16492	0,975119	15389	0,979524
25	11358	0,988017	12767	0,934181	18259	0,981795	22098	0,974708	12711	0,979482
26	11718	0,988017	148	0,933767	13197	0,981472	13770	0,974666	12882	0,979482
27	18009	0,987983	13206	0,931726	8029	0,981370	13513	0,974601	14689	0,979381
28	11007	0,987907	17512	0,931226	8030	0,981370	12665	0,974564	13043	0,979276
29	18831	0,987620	13961	0,931139	3955	0,981370	13823	0,974177	12766	0,979213
30	13102	0,984977	159	0,930731	18854	0,981126	12136	0,974117	13049	0,979114

Table A.6: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *bert-base-uncased* with chunking of protocol document and the second iteration of preprocessing.

A.7 SBERT: Truncation + Preprocessing 1

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	11860	1,000000	18877	1,000000
2	16597	0,827740	17712	0,999935	10353	1,000000	12551	1,000000	15182	0,970512
3	12361	0,818145	17711	0,994437	11119	0,999684	12710	1,000000	15183	0,963293
4	17421	0,814648	136	0,994347	11269	0,974502	12888	1,000000	14325	0,944657
5	12859	0,810505	237	0,917799	14507	0,947459	12757	0,989505	15025	0,924975
6	19454	0,809301	243	0,902329	10778	0,942447	12889	0,989505	15594	0,921211
7	16766	0,803189	242	0,902196	19115	0,834814	7557	0,825865	14429	0,917842
8	8145	0,800826	148	0,897978	14058	0,788456	12392	0,701756	14411	0,908359
9	6881	0,796858	149	0,897871	12679	0,785906	3493	0,700089	18409	0,898410
10	7111	0,796858	241	0,881183	6471	0,776943	3491	0,700089	17512	0,897876
11	14552	0,793174	12850	0,813351	6964	0,776943	3492	0,700089	16788	0,895878
12	10903	0,790055	12851	0,813351	18164	0,695715	15447	0,676768	8099	0,762861
13	14372	0,786883	12674	0,783822	17373	0,689869	15448	0,676768	18935	0,631955
14	12068	0,785235	12675	0,783822	17374	0,687348	15446	0,676768	17149	0,623578
15	16556	0,785209	6892	0,715532	3892	0,658125	15434	0,675778	19928	0,608483
16	16765	0,784321	6903	0,715532	3904	0,658125	15200	0,673599	11066	0,608483
17	16840	0,784321	7652	0,715532	7156	0,639835	16563	0,664898	11067	0,608483
18	6467	0,769369	8456	0,711619	7154	0,639835	15494	0,663350	11065	0,608483
19	15278	0,764086	17453	0,680377	17300	0,635850	13863	0,660132	20038	0,603773
20	18831	0,760303	17454	0,670144	17400	0,631182	11877	0,657554	17451	0,603773
21	18792	0,754219	16845	0,670042	17401	0,631182	13940	0,656813	20037	0,603773
22	18900	0,749452	12124	0,657188	18182	0,631182	3588	0,655624	17452	0,603773
23	18901	0,746980	12137	0,657188	14577	0,631018	12048	0,654365	15121	0,603773
24	19138	0,746777	8531	0,652621	18172	0,620996	3970	0,646516	15120	0,603773
25	18830	0,744793	8532	0,652621	17389	0,620958	3585	0,641243	15119	0,603773
26	16579	0,743475	8530	0,652621	17393	0,620958	3969	0,634720	9309	0,602148
27	7156	0,740842	9308	0,641766	6962	0,618182	3804	0,628254	9310	0,602148
28	7154	0,740842	26275	0,617047	3887	0,618182	14402	0,625471	11833	0,597296
29	19501	0,740826	90	0,604765	10903	0,613805	3254	0,624297	24005	0,592656
30	19500	0,740139	88	0,604765	16765	0,610589	3803	0,624297	17661	0,591869

Table A.7: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *all-mpnet-base-v2* with truncation of protocol document and the first iteration of preprocessing.

A.8 SBERT: Truncation + Preprocessing 2

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	17712	1,000000	8408	1,000000	11860	1,000000	18877	1,000000
2	16597	0,828063	137	1,000000	10353	1,000000	12551	1,000000	15182	0,960896
3	6467	0,773892	17711	0,991567	11119	0,998542	12710	1,000000	15183	0,960555
4	16766	0,767331	136	0,991567	11269	0,967031	12888	1,000000	14325	0,928768
5	12068	0,764594	237	0,986962	14507	0,942170	12757	0,976654	15025	0,910981
6	14552	0,761187	243	0,965287	10778	0,932745	12889	0,976654	15594	0,907224
7	19454	0,760200	242	0,965287	19115	0,846963	7557	0,898873	14429	0,900806
8	20008	0,758048	241	0,961000	13354	0,780749	3588	0,721107	14411	0,879152
9	16556	0,755235	12850	0,861480	12679	0,778670	12392	0,709014	17512	0,873120
10	17421	0,754910	12851	0,861480	14058	0,777466	3493	0,707916	16788	0,871481
11	19500	0,753790	12674	0,847255	6471	0,752625	3491	0,707916	18409	0,868327
12	19501	0,748563	12675	0,847255	6964	0,752625	3492	0,707916	8099	0,763249
13	15278	0,740661	148	0,844251	11148	0,719923	15200	0,702043	17149	0,660346
14	18831	0,739686	149	0,840990	11116	0,719153	3804	0,690308	18935	0,643448
15	16765	0,738730	8456	0,767475	3892	0,653852	15088	0,689914	9308	0,612038
16	16840	0,738730	6892	0,728169	3904	0,653852	15087	0,689914	20031	0,600561
17	12859	0,738625	6903	0,728169	17373	0,653824	14402	0,689720	19837	0,600561
18	14372	0,738505	7652	0,728169	11833	0,650860	14954	0,689620	18724	0,599653
19	12361	0,734504	11491	0,698419	17374	0,648926	15435	0,688532	12387	0,598361
20	18830	0,734377	16845	0,674789	18164	0,639429	11877	0,686714	19928	0,598036
21	16579	0,733831	14639	0,672359	6962	0,634013	15447	0,684615	11066	0,598036
22	13049	0,728348	14690	0,672359	3887	0,634013	15448	0,684615	11067	0,598036
23	13285	0,721606	17454	0,656028	14577	0,622175	15446	0,684615	11065	0,593525
24	12711	0,716147	9308	0,637678	16597	0,611366	14571	0,684002	12210	0,589079
25	12882	0,716147	12124	0,623593	15609	0,601412	16563	0,679798	12283	0,589079
26	13043	0,715712	12137	0,623593	13285	0,593936	15434	0,672878	12284	0,589079
27	18819	0,709089	3557	0,619679	14579	0,593798	12871	0,665235	12285	0,589079
28	18820	0,707181	3556	0,619679	15506	0,593132	12048	0,662799	12286	0,589079
29	17259	0,703532	2493	0,614970	17661	0,591776	13940	0,654619	13101	0,589079
30	17295	0,703532	90	0,612883	13994	0,589074	13863	0,651021	20038	0,585132

Table A.8: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *all-mpnet-base-v2* with truncation of protocol document and the second iteration of preprocessing.

A.9 SBERT: Chunking + Preprocessing 1

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	12888	1,000000	18877	1,000000
2	16766	0,984509	136	0,998290	10353	1,000000	12710	0,996601	16900	0,964059
3	12859	0,983036	17712	0,992316	11119	0,997652	12551	0,996439	16934	0,963165
4	12361	0,982973	17711	0,991714	19115	0,995225	12757	0,996324	17234	0,962177
5	17421	0,982720	237	0,959424	10778	0,995104	11860	0,995432	21861	0,954259
6	14372	0,982419	242	0,956077	15506	0,973149	12889	0,993609	17937	0,953143
7	16597	0,981811	243	0,954456	13354	0,961084	13056	0,906456	25850	0,952605
8	10903	0,979825	241	0,953099	11116	0,948964	13281	0,904685	18235	0,951045
9	14552	0,979487	17539	0,904045	11833	0,946711	13222	0,904234	12711	0,948973
10	8145	0,979055	17541	0,900142	11148	0,944852	13447	0,903881	12882	0,948973
11	18830	0,973816	8539	0,875273	17400	0,944653	13280	0,902991	13043	0,948499
12	18831	0,973040	148	0,875214	17401	0,944653	15494	0,896550	17995	0,947467
13	9183	0,970967	149	0,875212	18182	0,944653	11105	0,893592	18819	0,947140
14	10830	0,970218	8538	0,874424	17373	0,938771	20096	0,893316	18820	0,946257
15	16556	0,966416	8537	0,871539	18164	0,938758	15200	0,892190	17993	0,943047
16	6881	0,961068	8540	0,869168	17374	0,938410	11104	0,892115	13049	0,941090
17	7111	0,961068	23823	0,865657	17300	0,935103	15634	0,887686	19018	0,940008
18	19500	0,956487	23565	0,863780	20252	0,934355	15607	0,886456	18717	0,939019
19	15278	0,956109	8541	0,859828	20251	0,932691	15435	0,885218	19017	0,939005
20	16840	0,953931	12387	0,859605	20253	0,932537	15434	0,885191	13358	0,938999
21	16765	0,953908	6892	0,858986	18172	0,929708	11159	0,884983	19309	0,938651
22	16579	0,946964	6903	0,858986	17389	0,928052	13926	0,884756	18866	0,938342
23	19501	0,941759	7652	0,858986	17393	0,928052	11158	0,884512	18867	0,938342
24	18009	0,930624	12068	0,855664	18011	0,927200	11782	0,883815	18239	0,936560
25	18369	0,930477	19223	0,847050	15418	0,926363	15636	0,883363	19222	0,935073
26	18229	0,930351	11833	0,846307	17657	0,924242	14351	0,882801	20461	0,935004
27	18261	0,927030	15125	0,844288	18369	0,923851	16563	0,882455	20452	0,934758
28	13354	0,913931	15124	0,842793	13141	0,921062	12694	0,882080	19317	0,934654
29	21052	0,911432	12850	0,842035	13352	0,921062	11279	0,880821	16618	0,933154
30	7871	0,910485	12851	0,842035	13745	0,921062	26263	0,880725	15609	0,930758

Table A.9: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *all-mpnet-base-v2* with chunking of protocol document and the first iteration of preprocessing.

A.10 SBERT: Chunking + Preprocessing 2

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	12888	1,000000	18877	1,000000
2	16597	0,983526	136	0,994953	10353	1,000000	12710	0,997395	17937	0,959809
3	10903	0,982118	17711	0,991039	11119	0,997644	11860	0,996863	16934	0,953862
4	14552	0,980582	17712	0,986681	19115	0,994808	12551	0,996572	18235	0,952407
5	16766	0,980123	237	0,957399	10778	0,994176	12757	0,995820	13049	0,950397
6	18830	0,979746	243	0,946791	15506	0,954887	12889	0,994033	17234	0,946512
7	12859	0,978978	242	0,944168	13354	0,950761	20096	0,904008	18717	0,945985
8	12361	0,978897	241	0,937757	11116	0,928922	13447	0,887349	16900	0,945976
9	8145	0,978643	149	0,908473	11148	0,926205	15200	0,886906	18820	0,945917
10	14372	0,978635	148	0,898645	20253	0,924681	13281	0,883043	18819	0,945229
11	18831	0,978351	12387	0,869880	20252	0,924191	13222	0,882352	21861	0,945151
12	16556	0,976426	15125	0,867749	20251	0,922473	15494	0,881847	12711	0,942610
13	17421	0,973549	6892	0,866457	11833	0,921722	13280	0,881408	12882	0,942610
14	16579	0,967739	6903	0,866457	17400	0,915094	15435	0,880047	13043	0,940771
15	10830	0,966587	7652	0,866457	17401	0,915094	13056	0,879972	17995	0,937369
16	6881	0,965055	17541	0,866326	18182	0,915094	16563	0,878209	13358	0,935001
17	7111	0,965055	15124	0,859785	18369	0,915043	18153	0,876637	15609	0,933091
18	9183	0,962920	8456	0,857709	18261	0,914118	14351	0,876157	17993	0,932892
19	15278	0,957449	17539	0,853943	18229	0,911635	15434	0,873423	25850	0,931221
20	19500	0,956800	12850	0,851108	21052	0,911388	15636	0,873033	13994	0,930641
21	16765	0,954622	12851	0,851108	18164	0,906669	18152	0,872830	14313	0,930641
22	16840	0,954612	19223	0,846884	17740	0,905385	14786	0,868522	14344	0,930641
23	19501	0,932199	12674	0,846530	19500	0,905177	14787	0,868522	14597	0,930641
24	18369	0,927327	12675	0,846530	17300	0,904791	17597	0,866415	18239	0,929242
25	18261	0,923535	15466	0,842615	17374	0,904575	21872	0,863230	16618	0,928168
26	18229	0,920805	8538	0,838604	17373	0,904488	10338	0,862253	19018	0,926858
27	21052	0,920261	12068	0,833681	17743	0,904094	18151	0,861846	20452	0,924687
28	18009	0,917494	14552	0,833460	18009	0,903450	12465	0,860690	19017	0,924337
29	11007	0,907872	17421	0,832851	17742	0,901005	11660	0,859345	18866	0,920973
30	11315	0,904697	14372	0,832587	15418	0,900989	13770	0,858434	18867	0,920973

Table A.10: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *all-mpnet-base-v2* with chunking of protocol document and the second iteration of preprocessing.

A.11 Longformer: Preprocessing 1

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	12551	1,000000	18877	1,000000
2	14552	0,999673	17712	0,999995	10353	1,000000	11860	1,000000	15182	0,999839
3	16556	0,999663	136	0,999983	11119	0,999996	12710	1,000000	15183	0,999811
4	16597	0,999662	17711	0,999979	11269	0,999723	12888	1,000000	12210	0,999202
5	14372	0,999595	237	0,999892	16900	0,999506	12757	0,999810	12283	0,999202
6	12068	0,999557	243	0,999857	16934	0,999447	12889	0,999810	12284	0,999202
7	17421	0,999557	241	0,999852	14507	0,999389	7855	0,999233	12285	0,999202
8	12910	0,999548	242	0,999849	17498	0,999377	12392	0,999228	12286	0,999202
9	12361	0,999541	149	0,999524	12749	0,999377	3943	0,999212	13101	0,999202
10	18831	0,999521	148	0,999519	12445	0,999357	3944	0,999212	11057	0,999158
11	16877	0,999516	12797	0,998919	14070	0,999357	3925	0,999212	7569	0,999157
12	12445	0,999494	12674	0,998890	12182	0,999356	11660	0,999187	7588	0,999157
13	14070	0,999494	12675	0,998890	11858	0,999348	8544	0,999153	8016	0,999157
14	13102	0,999484	12850	0,998882	13358	0,999314	19460	0,999149	8074	0,999157
15	12859	0,999483	12851	0,998882	13102	0,999275	8543	0,999148	8075	0,999157
16	12955	0,999479	12796	0,998871	14963	0,999272	19005	0,999146	3883	0,999067
17	12182	0,999476	17539	0,998760	16563	0,999265	19117	0,999129	15263	0,999064
18	11858	0,999464	8456	0,998707	17812	0,999251	12765	0,999121	11481	0,999021
19	19500	0,999455	17541	0,998703	17234	0,999249	3493	0,999113	11262	0,999004
20	17498	0,999451	12387	0,998695	14962	0,999245	3491	0,999113	11482	0,998994
21	12749	0,999451	9308	0,998641	19153	0,999243	3492	0,999113	11159	0,998973
22	15434	0,999426	6892	0,998626	18007	0,999219	3807	0,999113	13076	0,998959
23	17740	0,999413	6903	0,998626	18831	0,999216	234	0,999111	11105	0,998932
24	13041	0,999401	7652	0,998626	14372	0,999204	2532	0,999111	17597	0,998927
25	14686	0,999397	16563	0,998552	12910	0,999199	19085	0,999110	14970	0,998918
26	12722	0,999395	15434	0,998532	15434	0,999191	12766	0,999108	11740	0,998909
27	20008	0,999390	15466	0,998513	21088	0,999190	3805	0,999107	15175	0,998890
28	12956	0,999389	11508	0,998501	18009	0,999187	3806	0,999107	7855	0,998878
29	18007	0,999376	19153	0,998488	12570	0,999187	3808	0,999107	14346	0,998874
30	12563	0,999372	16877	0,998468	14573	0,999187	15263	0,999102	14390	0,998864

Table A.11: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *longformer-base-4096* with the first iteration of preprocessing.

A.12 Longformer: Preprocessing 2

	Query doc									
	11508		137		10353		12888		18877	
	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity	Result doc	Similarity
1	11508	1,000000	137	1,000000	8408	1,000000	12551	1,000000	18877	1,000000
2	8145	0,999231	17712	1,000000	10353	1,000000	12710	1,000000	15182	0,999752
3	10903	0,999175	17711	0,999981	11119	0,999902	12888	1,000000	15183	0,999703
4	19501	0,999144	136	0,999981	12136	0,999004	11860	1,000000	14411	0,999299
5	11358	0,999133	237	0,999901	18009	0,998939	12757	0,999733	14325	0,999274
6	11718	0,999133	243	0,999900	15447	0,998883	12889	0,999733	15025	0,999229
7	18831	0,999106	242	0,999897	15448	0,998883	12392	0,998856	18409	0,999180
8	13049	0,999106	241	0,999729	15446	0,998883	17961	0,998856	15594	0,999147
9	12272	0,999014	12850	0,999548	6804	0,998881	17060	0,998854	16788	0,999132
10	13358	0,999012	12851	0,999548	6470	0,998881	8542	0,998793	14429	0,999120
11	9183	0,999005	12674	0,999548	3891	0,998881	10390	0,998767	17512	0,998905
12	13814	0,999005	12675	0,999548	19075	0,998866	3957	0,998766	12284	0,998694
13	17995	0,998995	8456	0,999105	13513	0,998859	3956	0,998766	12285	0,998694
14	11005	0,998994	6892	0,999030	12867	0,998857	3493	0,998763	12210	0,998694
15	6881	0,998992	6903	0,999030	13547	0,998857	3491	0,998763	12283	0,998694
16	7111	0,998992	7652	0,999030	21503	0,998821	3492	0,998763	12286	0,998694
17	19500	0,998986	17453	0,998958	17604	0,998815	3943	0,998684	13101	0,998694
18	15607	0,998981	14053	0,998922	18152	0,998813	3944	0,998684	22536	0,998690
19	11007	0,998981	18643	0,998898	17560	0,998787	3925	0,998684	19724	0,998538
20	16744	0,998978	18287	0,998891	17559	0,998774	14698	0,998678	13787	0,998538
21	12127	0,998977	12955	0,998758	21628	0,998757	14696	0,998678	14123	0,998538
22	17755	0,998974	13206	0,998724	18151	0,998750	14505	0,998622	14204	0,998538
23	13770	0,998973	9308	0,998710	11874	0,998730	14504	0,998606	14414	0,998538
24	12128	0,998971	12563	0,998704	15200	0,998720	9298	0,998595	14287	0,998503
25	17498	0,998964	8538	0,998695	14351	0,998712	9297	0,998595	14285	0,998503
26	12749	0,998964	8537	0,998678	7557	0,998710	9296	0,998595	22472	0,998480
27	12570	0,998963	13141	0,998631	15389	0,998706	18475	0,998561	20286	0,998473
28	14573	0,998963	13352	0,998631	3949	0,998677	15175	0,998475	22428	0,998471
29	12863	0,998963	13745	0,998631	3950	0,998677	19117	0,998473	20942	0,998470
30	13591	0,998937	8540	0,998626	3952	0,998677	20765	0,998470	11057	0,998443

Table A.12: The search results for top-30 most similar protocol documents, and their similarity scores, retrieved for query documents 11508, 137, 10353, 12888 and 18877 using *longformer-base-4096* with the second iteration of preprocessing.

B

Appendix B - Jaccard similarity

B.1 Query document 11508

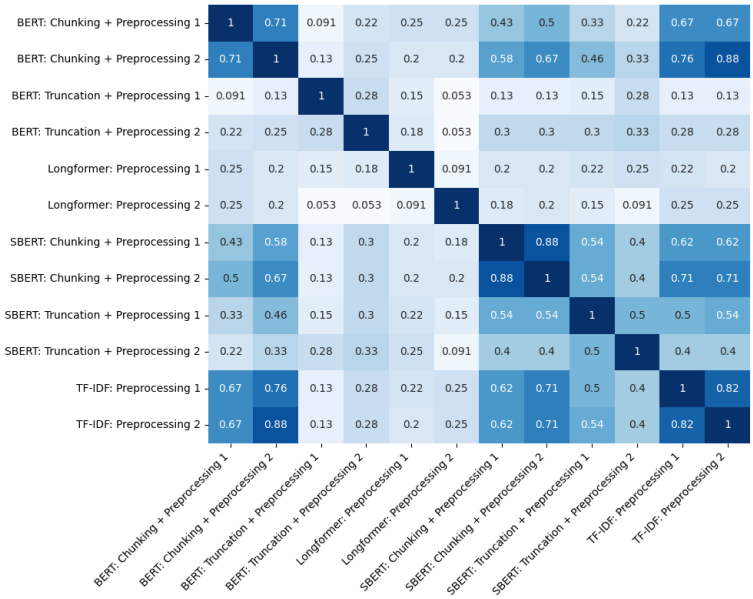


Figure B.1: A heatmap visualising the overlap between search results from the different models used for query document 11508.

B.2 Query document 137

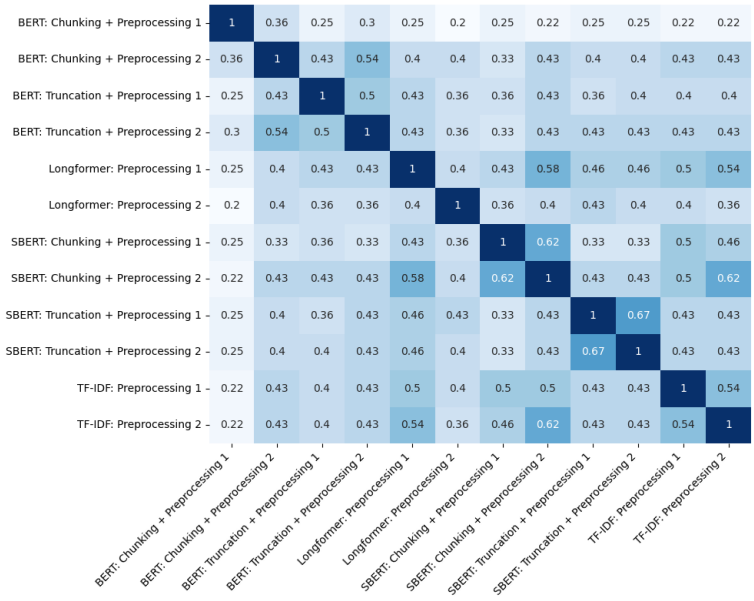


Figure B.2: A heatmap visualising the overlap between search results from the different models used for query document 137.

B.3 Query document 10353

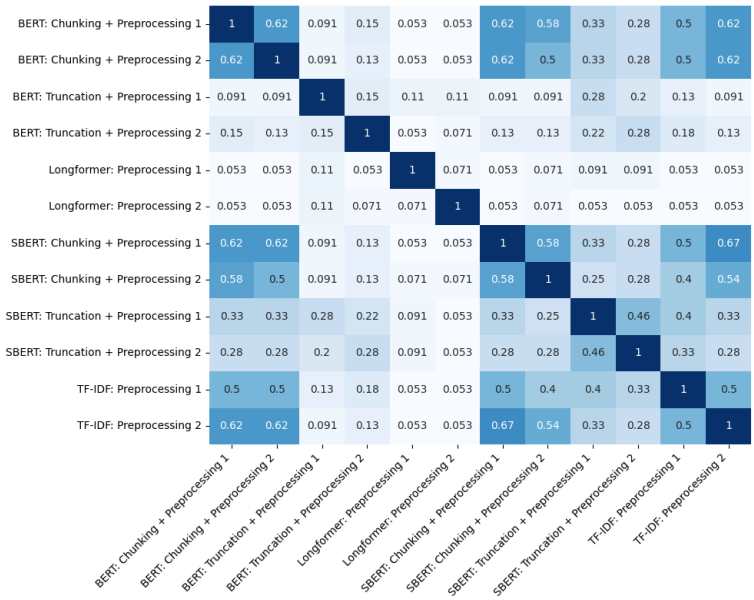


Figure B.3: A heatmap visualising the overlap between search results from the different models used for query document 10353.

B.4 Query document 12888

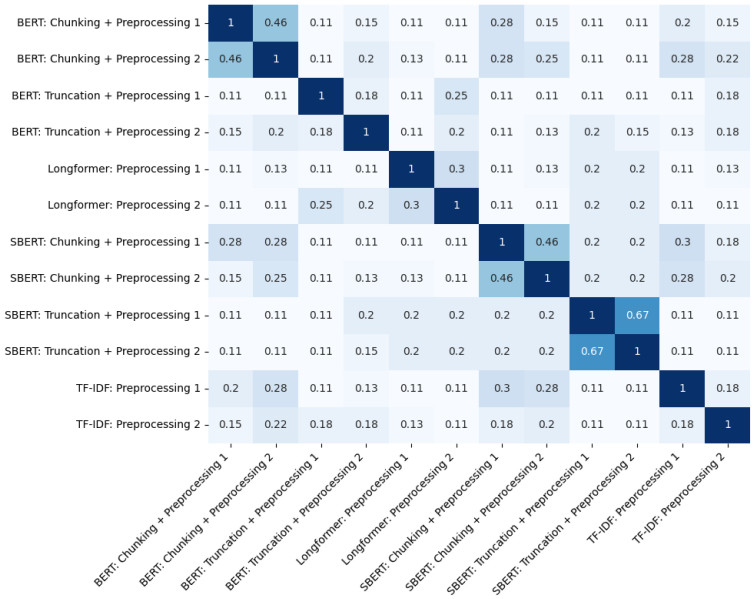


Figure B.4: A heatmap visualising the overlap between search results from the different models used for query document 12888.

B.5 Query document 18877

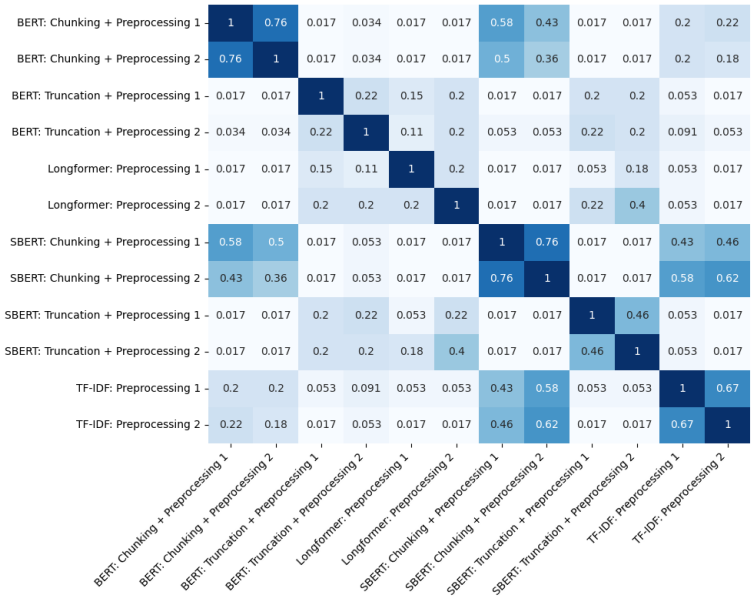


Figure B.5: A heatmap visualising the overlap between search results from the different models used for query document 18877.

C

Appendix C - Manual evaluation (Query document 11508)

C.1 TF-IDF: Preprocessing 1

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"Interesting"
11508	1,000000	1	1	1	1	0
18830	0,916554	1	1	1	1	0
18831	0,914864	1	1	1	1	0
12361	0,914085	1	1	1	1	0
12859	0,913604	1	1	1	1	0
16766	0,913400	1	1	1	1	0
17421	0,912690	1	1	1	1	0
16597	0,910889	1	1	1	1	0
14372	0,907303	1	1	1	1	0
14552	0,906646	1	1	1	1	0
16556	0,905169	1	1	1	1	0
16579	0,898800	1	1	1	1	0
10903	0,877005	1	1	1	1	0
16840	0,868747	1	1	1	1	1
16765	0,868486	1	1	1	1	1
15278	0,868024	1	1	1	1	1
19500	0,829629	1	1	1	1	1
19501	0,818377	1	1	1	1	1
11717	0,782407	1	1	1	1	1
11315	0,782407	1	1	1	1	1
11007	0,781105	1	1	1	1	1
11718	0,778699	1	1	1	1	1
11358	0,778699	1	1	1	1	1
18009	0,760440	1	1	1	1	1
10830	0,747803	1	1	1	1	1
12271	0,745581	1	1	1	1	1
9183	0,744462	1	1	1	1	1
6881	0,744146	1	1	1	1	1
7111	0,744146	1	1	1	1	1
17498	0,743587	1	1	1	1	1

Table C.1: Manual evaluation of search results for query document 11508 from the TF-IDF model with preprocessing 1.

C.2 TF-IDF: Preprocessing 2

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
12361	0,707436	1	1	1	1	0
12859	0,706620	1	1	1	1	0
16766	0,706379	1	1	1	1	0
17421	0,706378	1	1	1	1	0
18831	0,702670	1	1	1	1	0
10903	0,690713	1	1	1	1	0
14372	0,660305	1	1	1	1	0
16840	0,653334	1	1	1	1	1
16765	0,653313	1	1	1	1	1
15278	0,653143	1	1	1	1	1
16597	0,650147	1	1	1	1	0
14552	0,643564	1	1	1	1	0
18830	0,623548	1	1	1	1	0
16556	0,606051	1	1	1	1	0
16579	0,599198	1	1	1	1	0
10830	0,584017	1	1	1	1	1
9183	0,581511	1	1	1	1	1
6881	0,581254	1	1	1	1	1
7111	0,581254	1	1	1	1	1
19500	0,563131	1	1	1	1	1
19501	0,545425	1	1	1	1	1
8145	0,507003	1	1	1	1	1
11007	0,488079	1	1	1	1	1
11717	0,488030	1	1	1	1	1
11315	0,488030	1	1	1	1	1
11718	0,487874	1	1	1	1	1
11358	0,487874	1	1	1	1	1
10283	0,480535	1	1	1	1	1
10410	0,480535	1	1	1	1	1

Table C.2: Manual evaluation of search results for query document 11508 from the TF-IDF model with preprocessing 2.

C.3 BERT: Truncation + Preprocessing 1

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16579	0,965460	1	1	1	1	0
21004	0,962400	1	0	0	1	0
22472	0,960640	1	0	0	1	0
20578	0,960425	0	0	0	0	0
16597	0,960397	1	1	1	1	0
18012	0,958937	1	0	0	1	0
6467	0,958055	1	1	1	1	1
16556	0,957960	1	1	1	1	0
20577	0,956586	0	0	0	0	0
13049	0,953065	1	1	1	1	1
14552	0,952289	1	1	1	1	0
12361	0,951030	1	1	1	1	0
10393	0,948520	0	0	0	0	0
10391	0,948520	0	0	0	0	0
15278	0,948034	1	1	1	1	1
15609	0,947943	0	1	1	0	1
12711	0,947932	0	1	1	0	1
12882	0,947932	1	1	1	1	1
13043	0,947307	1	1	1	1	1
20008	0,947122	1	1	1	1	1
13767	0,947059	0	0	0	0	0
14067	0,947059	0	0	0	0	0
12722	0,946204	1	1	1	1	1
17387	0,945626	1	0	0	0	0
14825	0,945178	0	0	0	0	0
12563	0,944818	1	1	1	1	1
17755	0,944596	1	1	1	1	1
11355	0,944365	0	0	0	0	0
13141	0,944101	0	0	0	0	0

Table C.3: Manual evaluation of search results for query document 11508 from the BERT model with truncation and preprocessing 1.

C.4 BERT: Truncation + Preprocessing 2

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"Interesting"
11508	1,000000	1	1	1	1	0
12361	0,978312	1	1	1	1	0
12859	0,977213	1	1	1	1	0
14552	0,973407	1	1	1	1	0
15278	0,969506	1	1	1	1	1
16597	0,966222	1	1	1	1	0
18012	0,965736	1	0	0	1	0
2972	0,965167	0	0	0	0	0
17421	0,963620	1	1	1	1	0
21004	0,963461	1	0	0	1	0
19501	0,963249	1	1	1	1	1
22472	0,962649	1	0	0	1	0
20578	0,962450	0	0	0	0	0
19454	0,962082	0	1	1	1	1
18831	0,961521	1	1	1	1	0
16579	0,960168	1	1	1	1	0
16766	0,959430	1	1	1	1	0
20008	0,959345	1	1	1	1	1
20578	0,959071	0	0	0	0	0
9308	0,958037	0	0	0	0	0
20577	0,957828	0	0	0	0	0
14372	0,956639	1	1	1	1	0
3431	0,956548	0	0	0	0	0
13141	0,956079	0	0	0	0	0
13352	0,956079	1	0	0	0	0
13745	0,956079	1	0	0	0	1
16893	0,954763	0	0	0	0	0
16583	0,954199	0	0	0	0	0
16892	0,951596	0	0	0	0	0
18229	0,951085	0	0	0	0	0

Table C.4: Manual evaluation of search results for query document 11508 from the BERT model with truncation and preprocessing 2.

C.5 BERT: Chunking + Preprocessing 1

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16766	0,994987	1	1	1	1	0
10903	0,994560	1	1	1	1	0
12859	0,994306	1	1	1	1	0
12361	0,994298	1	1	1	1	0
14552	0,993986	1	1	1	1	0
17421	0,993830	1	1	1	1	0
14372	0,993811	1	1	1	1	0
18831	0,993638	1	1	1	1	0
16597	0,993427	1	1	1	1	0
8145	0,993061	1	1	1	1	1
18830	0,992692	1	1	1	1	0
10283	0,992560	1	1	1	1	1
10410	0,992560	1	1	1	1	1
15278	0,991802	1	1	1	1	1
6881	0,991260	1	1	1	1	1
7111	0,991260	1	1	1	1	1
18009	0,990893	1	1	1	1	1
10830	0,990861	1	1	1	1	1
9183	0,990836	1	1	1	1	1
11007	0,990470	1	1	1	1	1
11358	0,989965	1	1	1	1	1
11718	0,989965	1	1	1	1	1
11315	0,989710	1	1	1	1	1
11717	0,989710	1	1	1	1	1
12182	0,989383	1	1	1	1	1
11858	0,988910	1	1	1	1	1
12271	0,988666	1	1	1	1	1
17498	0,988478	1	1	1	1	1
12749	0,988478	1	1	1	1	1

Table C.5: Manual evaluation of search results for query document 11508 from the BERT model with chunking and preprocessing 1.

C.6 BERT: Chunking + Preprocessing 2

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16597	0,993628	1	1	1	1	0
10903	0,992609	1	1	1	1	0
12361	0,991734	1	1	1	1	0
12859	0,991436	1	1	1	1	0
18830	0,991406	1	1	1	1	0
14372	0,991268	1	1	1	1	0
16766	0,990995	1	1	1	1	0
8145	0,990455	1	1	1	1	1
17421	0,989953	1	1	1	1	0
10283	0,989779	1	1	1	1	1
10410	0,989779	1	1	1	1	1
10830	0,989778	1	1	1	1	1
14552	0,989584	1	1	1	1	0
16840	0,989321	1	1	1	1	1
16765	0,989316	1	1	1	1	1
16579	0,989197	1	1	1	1	0
15278	0,989013	1	1	1	1	1
6881	0,988750	1	1	1	1	1
7111	0,988750	1	1	1	1	1
9183	0,988745	1	1	1	1	1
16556	0,988628	1	1	1	1	0
11315	0,988128	1	1	1	1	1
11717	0,988128	1	1	1	1	1
11358	0,988017	1	1	1	1	1
11718	0,988017	1	1	1	1	1
18009	0,987983	1	1	1	1	1
11007	0,987907	1	1	1	1	1
18831	0,987620	1	1	1	1	0
13102	0,984977	1	0	1	0	0

Table C.6: Manual evaluation of search results for query document 11508 from the BERT model with chunking and preprocessing 2.

C.7 SBERT: Truncation + Preprocessing 1

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16597	0,827740	1	1	1	1	0
12361	0,818145	1	1	1	1	0
17421	0,814648	1	1	1	1	0
12859	0,810505	1	1	1	1	0
19454	0,809301	0	1	1	1	1
16766	0,803189	1	1	1	1	0
8145	0,800826	1	1	1	1	1
6881	0,796858	1	1	1	1	1
7111	0,796858	1	1	1	1	1
14552	0,793174	1	1	1	1	0
10903	0,790055	1	1	1	1	0
14372	0,786883	1	1	1	1	0
12068	0,785235	1	1	1	1	1
16556	0,785209	1	1	1	1	0
16765	0,784321	1	1	1	1	1
16840	0,784321	1	1	1	1	1
6467	0,769369	1	1	1	1	1
15278	0,764086	1	1	1	1	1
18831	0,760303	1	1	1	1	0
18792	0,754219	1	0	0	0	0
18900	0,749452	1	1	0	0	1
18901	0,746980	1	1	0	0	1
19138	0,746777	1	1	0	0	1
18830	0,744793	1	1	1	1	0
16579	0,743475	1	1	1	1	0
7156	0,740842	0	0	0	0	0
7154	0,740842	0	0	1	0	0
19501	0,740826	1	1	1	1	1
19500	0,740139	1	1	1	1	1

Table C.7: Manual evaluation of search results for query document 11508 from the SBERT model with truncation and preprocessing 1.

C.8 SBERT: Truncation + Preprocessing 2

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16597	0,828063	1	1	1	1	0
6467	0,773892	1	1	1	1	1
16766	0,767331	1	1	1	1	0
12068	0,764594	1	1	1	1	1
14552	0,761187	1	1	1	1	0
19454	0,760200	0	1	1	1	1
20008	0,758048	1	1	1	1	1
16556	0,755235	1	1	1	1	0
17421	0,754910	1	1	1	1	0
19500	0,753790	1	1	1	1	1
19501	0,748563	1	1	1	1	1
15278	0,740661	1	1	1	1	1
18831	0,739686	1	1	1	1	0
16765	0,738730	1	1	1	1	1
16840	0,738730	1	1	1	1	1
12859	0,738625	1	1	1	1	0
14372	0,738505	1	1	1	1	0
12361	0,734504	1	1	1	1	0
18830	0,734377	1	1	1	1	0
16579	0,733831	1	1	1	1	0
13049	0,728348	1	1	1	1	1
13285	0,721606	0	0	0	0	0
12711	0,716147	0	1	1	0	1
12882	0,716147	1	1	1	1	1
13043	0,715712	1	1	1	1	1
18819	0,709089	0	0	1	0	0
18820	0,707181	0	0	1	0	0
17259	0,703532	1	0	1	0	0
17295	0,703532	1	0	1	0	0

Table C.8: Manual evaluation of search results for query document 11508 from the SBERT model with truncation and preprocessing 2.

C.9 SBERT: Chunking + Preprocessing 1

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16766	0,984509	1	1	1	1	0
12859	0,983036	1	1	1	1	0
12361	0,982973	1	1	1	1	0
17421	0,982720	1	1	1	1	0
14372	0,982419	1	1	1	1	0
16597	0,981811	1	1	1	1	0
10903	0,979825	1	1	1	1	0
14552	0,979487	1	1	1	1	0
8145	0,979055	1	1	1	1	1
18830	0,973816	1	1	1	1	0
18831	0,973040	1	1	1	1	0
9183	0,970967	1	1	1	1	1
10830	0,970218	1	1	1	1	1
16556	0,966416	1	1	1	1	0
6881	0,961068	1	1	1	1	1
7111	0,961068	1	1	1	1	1
19500	0,956487	1	1	1	1	1
15278	0,956109	1	1	1	1	1
16840	0,953931	1	1	1	1	1
16765	0,953908	1	1	1	1	1
16579	0,946964	1	1	1	1	0
19501	0,941759	1	1	1	1	1
18009	0,930624	1	1	1	1	1
18369	0,930477	0	0	1	0	0
18229	0,930351	0	0	0	0	0
18261	0,927030	0	0	1	0	0
13354	0,913931	0	0	0	0	0
21052	0,911432	1	0	0	0	1
7871	0,910485	0	0	0	0	0

Table C.9: Manual evaluation of search results for query document 11508 from the SBERT model with chunking and preprocessing 1.

C.10 SBERT: Chunking + Preprocessing 2

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
16597	0,983526	1	1	1	1	0
10903	0,982118	1	1	1	1	0
14552	0,980582	1	1	1	1	0
16766	0,980123	1	1	1	1	0
18830	0,979746	1	1	1	1	0
12859	0,978978	1	1	1	1	0
12361	0,978897	1	1	1	1	0
8145	0,978643	1	1	1	1	1
14372	0,978635	1	1	1	1	0
18831	0,978351	1	1	1	1	0
16556	0,976426	1	1	1	1	0
17421	0,973549	1	1	1	1	0
16579	0,967739	1	1	1	1	0
10830	0,966587	1	1	1	1	1
6881	0,965055	1	1	1	1	1
7111	0,965055	1	1	1	1	1
9183	0,962920	1	1	1	1	1
15278	0,957449	1	1	1	1	1
19500	0,956800	1	1	1	1	1
16765	0,954622	1	1	1	1	1
16840	0,954612	1	1	1	1	1
19501	0,932199	1	1	1	1	1
18369	0,927327	0	0	1	0	0
18261	0,923535	0	0	1	0	0
18229	0,920805	0	0	0	0	0
21052	0,920261	1	0	0	0	1
18009	0,917494	1	1	1	1	1
11007	0,907872	1	1	1	1	1
11315	0,904697	1	1	1	1	1

Table C.10: Manual evaluation of search results for query document 11508 from the SBERT model with chunking and preprocessing 2.

C.11 Longformer: Preprocessing 1

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
14552	0,999673	1	1	1	1	0
16556	0,999663	1	1	1	1	0
16597	0,999662	1	1	1	1	0
14372	0,999595	1	1	1	1	0
12068	0,999557	1	1	1	1	1
17421	0,999557	1	1	1	1	0
12910	0,999548	0	0	1	0	0
12361	0,999541	1	1	1	1	0
18831	0,999521	1	1	1	1	0
16877	0,999516	0	0	0	0	0
12445	0,999494	1	0	1	0	0
14070	0,999494	1	0	1	0	0
13102	0,999484	1	0	1	0	0
12859	0,999483	1	1	1	1	0
12955	0,999479	1	0	1	0	0
12182	0,999476	1	1	1	1	1
11858	0,999464	1	1	1	1	1
19500	0,999455	1	1	1	1	1
17498	0,999451	1	1	1	1	1
12749	0,999451	1	1	1	1	1
15434	0,999426	0	0	0	0	0
17740	0,999413	1	0	0	0	0
13041	0,999401	1	0	1	0	0
14686	0,999397	1	0	0	0	0
12722	0,999395	1	1	1	1	1
20008	0,999390	1	1	1	1	1
12956	0,999389	1	0	1	0	0
18007	0,999376	0	0	1	0	0
12563	0,999372	1	1	1	1	1

Table C.11: Manual evaluation of search results for query document 11508 from the Longformer model with preprocessing 1.

C.12 Longformer: Preprocessing 2

Protocol	Similarity score	Target	Methodology	Readout	Reagent	"interesting"
11508	1,000000	1	1	1	1	0
8145	0,999231	1	1	1	1	1
10903	0,999175	1	1	1	1	0
19501	0,999144	1	1	1	1	1
11358	0,999133	1	1	1	1	1
11718	0,999133	1	1	1	1	1
18831	0,999106	1	1	1	1	0
13049	0,999106	1	1	1	1	1
12272	0,999014	1	0	1	0	0
13358	0,999012	1	0	1	0	0
9183	0,999005	1	1	1	1	1
13814	0,999005	0	0	0	0	0
17995	0,998995	0	0	1	0	0
11005	0,998994	0	0	0	0	0
6881	0,998992	1	1	1	1	1
7111	0,998992	1	1	1	1	1
19500	0,998986	1	1	1	1	1
15607	0,998981	0	0	1	0	0
11007	0,998981	1	1	1	1	1
16744	0,998978	1	0	0	0	0
12127	0,998977	0	0	0	0	0
17755	0,998974	1	1	1	1	1
13770	0,998973	0	0	0	0	0
12128	0,998971	0	0	0	0	0
17498	0,998964	1	1	1	1	1
12749	0,998964	1	1	1	1	1
12570	0,998963	0	0	0	0	0
14573	0,998963	1	0	1	0	0
12863	0,998963	0	0	0	0	0
13591	0,998937	0	0	1	0	0

Table C.12: Manual evaluation of search results for query document 11508 from the Longformer model with preprocessing 2.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY