JOHANNES KEPLER
UNIVERSITÄT LINZ

Submitted by
Luu Ngoc Tram - K11947734

Submitted at
Institute for Business Informatics - Data & Knowledge Engineering

Supervisor
Assoz.-Prof. Mag. Dr. Christoph Schütz

Co-Supervisor
Simon Staudinger, MSc

August 2023

A recommender system for research papers in medicine at Bloom Diagnostics

Master Thesis
to obtain the academic degree of
Master of Science
in the Master's Program
Economic & Business Analytics

JOHANNES KEPLER UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at

Abstract

The medical field is rapidly evolving due to advancements in Artifical Intelligence, which allows machines to perform cognitive activities to achieve specific objectives using data as input. As a result, there is a growing demand for text-mining methods to extract useful insights from vast volumes of medical textual data. However, the application of Natural Language Processing techniques in medicine faces several challenges, including the need to adapt to medical terminologies and the differences between ordinary corpora and medical corpora. Deep learning approaches have made advances in text mining methods feasible, but they still face challenges such as the difficulties of scaling efficiently and the lack of domain-specific data.

An actual use case of Bloom Diagnostics GmbH, a start-up in the field of digital health that offers home access to blood tests and health advice, served as the basis for the thesis. In order to validate suggestions to users provided based on the blood tests, the medical team must review a vast amount of relevant scientific records to ensure accurate suggestions to their users. A systematic literature search process is required by the team to extract valuable information, which is time-consuming and manual. With the purpose of accelerating the literature search process, the thesis proposes a general concept for building a recommender system for research papers in the medical field, and focuses on ranking/ re-ranking passages based on their relevance to natural language questions.

Various NLP models, including BM25, BERT, and BioBERT, are compared to determine the most efficient setting. Apart from BM25 and BERT, which can already perform ranking/ re-ranking tasks, BioBERT needs to be fine-tuned separately with medical data so the model is comparable with other ones. By comparing and contrasting those models in different combinations and settings and evaluating the results using various criteria, the main contribution of this thesis is a solution that helps researchers save time by recommending the most relevant passages returned by these techniques. The recommender system, which focuses on recommendations of passages relevant for natural language questions, is built using the current literature search web application for medical researchers at Bloom Diagnostics.

Zusammenfassung

Das medizinische Fachgebiet entwickelt sich aufgrund von Fortschritten in der Künstlichen Intelligenz (KI) rasant weiter, die Maschinen befähigt, kognitive Aktivitäten zur Erreichung spezifischer Ziele unter Verwendung von Daten als Eingabe durchzuführen. Dies hat zur Folge, dass eine wachsende Nachfrage nach Textmining-Methoden besteht, um nützliche Erkenntnisse aus umfangreichen Mengen medizinischer Textdaten zu extrahieren. Die Anwendung von Techniken zur natürlichen Sprachverarbeitung in der Medizin steht jedoch vor mehreren Herausforderungen, darunter die Notwendigkeit, sich an medizinische Fachterminologie anzupassen sowie die Unterschiede zwischen gewöhnlichen Korpora und medizinischen Korpora. Obwohl Deep-Learning-Ansätze Fortschritte in Textmining-Methoden ermöglicht haben, bestehen nach wie vor Herausforderungen wie die effiziente Skalierung und der Mangel an domänenspezifischen Daten.

Ein konkreter Anwendungsfall der Bloom Diagnostics GmbH, eines Start-ups im Bereich der digitalen Gesundheit, das Bluttests und Gesundheitsberatung für zu Hause anbietet, diente als Grundlage für die vorliegende Arbeit. Um Vorschläge für Benutzer aufgrund von Bluttestergebnissen zu validieren, muss das medizinische Team eine umfangreiche Menge relevanter wissenschaftlicher Aufzeichnungen überprüfen, um genaue Empfehlungen für die Benutzer sicherzustellen. Ein systematischer Literaturrechercheprozess ist erforderlich, um wertvolle Informationen zu extrahieren, was zeitaufwändig und manuell ist. Mit dem Ziel, den Literaturrechercheprozess zu beschleunigen, schlägt die Arbeit ein allgemeines Konzept für den Aufbau eines Empfehlungssystems für medizinische Forschungsarbeiten vor und konzentriert sich auf das Ranking/Re-Ranking von Passagen basierend auf ihrer Relevanz für Fragen natürlicher Sprache.

Verschiedene Modelle für die Verarbeitung natürlicher Sprache, einschließlich BM25, BERT und BioBERT, werden verglichen, um die effizienteste Einstellung zu ermitteln. BioBERT muss neben BM25 und BERT, die bereits Aufgaben des Rankings und Re-Rankings durchführen können, separat mit medizinischen Daten feinabgestimmt werden, um das Modell mit anderen vergleichbar zu machen. Durch den Vergleich und die Kontrastierung dieser Modelle in verschiedenen Kombinationen und Einstellungen sowie die Bewertung der Ergebnisse anhand verschiedener Kriterien besteht der Hauptbeitrag dieser Arbeit in einer Lösung, die Forschern hilft, Zeit zu sparen, indem sie die relevantesten Passagen empfiehlt, die durch diese Techniken zurückgegeben werden. Das Empfehlungssystem, das sich auf Empfehlungen von für natürliche Sprachfragen relevante Passagen konzentriert, wird mithilfe der aktuellen Literaturrecherchewebanwendung für medizinische Forscher bei Bloom Diagnostics erstellt.

Contents

1 Introduction

Our healthcare systems are being revolutionized and reshaped by Artificial Intelligence (AI), which is the capacity of machines to do cognitive activities to reach a specific objective using data input [34]. In tandem with the advancements of data mining methods in the medical domain, the volume of data increases dramatically. Those medical data are typically vast and diverse, including images, patient records, laboratory data, and the doctor's diagnoses [17]. Among those, large volumes of documented biological scientific facts and knowledge exist in text format [15]. Medical reports, research, books and guidelines, biological databases, patient records, and web content are common channels for the dissemination of scientific textual data [36]. A huge amount of medical textual data exists on the internet in different format that can potentially provide useful information [17]. As a result, there is a growing demand for text-mining methods to extract useful insights from those text data.

The broad availability of text mining methods for data analysis necessitates that researchers select the most efficient approach for addressing medical problems [12]. However, there are limits to directly applying state-of-the-art Natural Language Processing (NLP) approaches to the medical field. The vast amount of medical terminology requires text mining methods to adapt to terms specific to the medical domain for improved comprehension and analysis of these texts. In addition, there are significant differences between the word distributions of ordinary corpora and medical corpora, which might be problematic for text mining algorithms. Consequently, domain-specific text mining methods must be developed [41]. Among those methods, deep learning approaches employed in NLP have made advances in text mining methods in the medical domain feasible.

In spite of the progress, the application of NLP techniques in medicine still faces numerous obstacles. Deep learning is difficult to scale efficiently as neural network models require a large computational cost of time, processor power, and memory to train and implement [16]. Second, the lack of medical datasets impedes the development of mining methods for this domain.

This thesis describes an actual use case of Bloom Diagnostics GmbH. As a digital health start-up that provides blood tests for everyone to access at home, the firm must guarantee the accuracy of test findings and give proof of their diagnosis. To correctly provide clients with recommendations regarding their health status based on blood test results, a team of medical professionals must review a vast quantity of relevant scientific records, taking into account age, gender, ethnicity, and other factors. A process of systematic literature search is required to provide evidence for those recommendations. The majority of those literature searches are performed manually, meaning that researchers must go through each document to select and extract valuable information. This results in a complex and time-consuming process for the medical team, which desires a solution to help accelerate this process.

By setting up evaluations for various NLP models with a real-world use case at Bloom Diagnostics, the objective of this thesis is to compare those models in order to select the most efficient models that help reduce the time and effort required for the literature search process. This thesis first proposes a general concept for building a recommender system for research papers, adapted from related works, and then focuses on ranking/re-ranking of passages in documents according to their relevance to the natural language question. During ranking/re-ranking, the most relevant passages are prioritized, presenting them at the top of the search pages. This enables researchers to quickly scan through these passages instead of having to go through all the documents.

The thesis compares various NLP techniques for document or passage ranking/re-ranking, including

BM25, BERT, and its variant BioBERT. Various settings are evaluated, including the use of BM25 alone and the combination of BM25 with BERT and BioBERT, to figure out the most efficient setting. The thesis leverages a research study by Nogueira et al. [50] for combining BM25 and BERT in ranking passages. In order to utilize BioBERT, a new dataset was generated for fine-tuning BioBERT for passage re-ranking. Different settings are evaluated in terms of execution time, F1 score, and other measures, using actual data sets from Bloom Diagnostics.

By experimenting with various NLP techniques, namely BM25, BERT and BioBERT, in a variety of combinations among those models, such as the use of BM25 and BERT or BM25 and BioBERT, the main contribution of this thesis is a solution that helps researchers save time by recommending the most relevant passages returned by those techniques. The solution should have a balance between efficiency, effectiveness and usability and be applicable to the current situation at Bloom Diagnostics. In addition, by utilizing a real-world dataset for evaluation, the thesis aims to provide an additional perspective on the effectiveness of the investigated techniques in a real-world situation.

The passage re-ranking by BM25, BERT, or BioBERT is the last step in the process followed by the recommender system. By definition, "recommender system" is a fully operational software system which uses at least one framework to generate suggestions [11]. Recommender systems include a user interface, a list of candidate recommendations, and operators who own and operate the system [11]. To build a recommender system for medical literature papers, the user interface can be utilized from the current literature search web application for medical researchers at Bloom Diagnostics developed by the author of this thesis. The thesis concentrates on recommendations of passages relevant for natural language questions.

The structure of the thesis is as follows. Chapter 2 presents some essential works and the basis of all approaches and methodologies utilized in the thesis. Chapter 3 describes the present situation at the company that necessitated the development of a recommender system for medical research papers and its characteristics. Chapter 4 provides a general method to build a recommender system as well as the main steps to preprocess data and fine-tune BioBERT. Chapter 5 then describes in greater detail several use cases, the execution of the experiments and the results of the evaluation for those use cases, followed by Chapter 6 of discussions, limitations of the current approach, and some future work.

2 Background

This chapter provides an overview of prior work in the fields of medical research paper recommender systems. The thesis will next examine the concepts of some of the most common models to assist in the development of a research paper recommender system tailored to our needs, namely BM25, BERT and BioBERT.

2.1 Research paper recommender system in medical field

In their literature review, Beel et al. note that since 2000, more than 200 research articles on recommender systems for research papers have been published [11]. In the last two decades, recommender systems have proliferated as a result of the need for people in various fields to search for information. Among these, the most notable are Google Scholar, PubMed, ResearchGate, CiteULike or BibTip [11]. Depending on objectives, there are numerous methodologies, such as content-based, collaborative, or graph-based filtering, and evaluation methods, ranging from offline/online evaluation to quantitative and qualitative user studies [11].

In medical field, PubMed has been one of the primary resources for research paper recommender systems in particular and retrieval of medical literature in general. As the thesis also uses the database of PubMed, we will only focus on PubMed-based recommender systems for medical research papers. PubMed is the premier health sciences research database, which is produced by the National Center for Biotechnology Information (NCBI) belong to the United States National Library of Medicine (NLM). It is utilized by researchers, practitioners, and students worldwide. PubMed website provides different information resources including MEDLINE, PubMed Central, and a selection of free e-books on health sciences themes [72]. PubMed contains more than 30 million citations, originating from more than 5,300 publications currently indexed in MEDLINE and over 3,000 journals that have deposited content in PubMed Central [47].

Using "PubMed and Beyond: A Survey of Web-Based Tools for Searching Biomedical Literature" [44] and all the systems given in this article as a starting point, snowballing technique [73] is then utilized to identify other relevant systems. In the following table, 42 medical literature search engines are presented in total.

Table 2.1: List of examined literature recommender systems in biomedicine

System	Year	Main features	Methodology & database	Accessible website
MEDSUM [13]	1986	indicating when the first MEDLINE-indexed articles on the subject were published		No
PubCrawler [37]	1999	notifying users of new articles based on previously saved searches		`https://pubcrawler.gen.tcd.ie/`

XplorMed [53]	2001	recommending primary groups of connected subjects and publications, avoiding the user from having to read all abstracts	MeSH, fuzzy sets, part-of-speech tagging	No
askMEDLINE [31]	2005	accepting free-text, natural language queries and locating relevant MEDLINE/PubMed citations without specialist vocabularies	Multi-round searches, MeSH	`https://pubmedhh.nlm.nih.gov/ask/index.php`
BioGyan	2005	database-based exhaustive combinatorial searching for a gene/protein, cell type-/organism, or topic; retrieve and sort abstractions; emphasize pertinent text; see paths and three-dimensional structures		No
GoPubMed [26]	2005	grouping PubMed abstracts according to MeSH or Gene Ontology (GO) keywords	term extraction algorithm, MeSH, GO	No
SLIM [45]	2005	interactive sliders in the search form governed search settings including limits, filters, and MeSH terms	Entrez Programming Utilities (E-Utilities)	No
BioIE [25]	2005	extracting informative statements from biomedical literature regarding protein families, their structures, functions, and illnesses	MeSH, N-gram distribution	`http://umber.sbs.man.ac.uk/dbbrowser/bioie/`
McSyBi [75]	2006	displaying results as hierarchical or non-hierarchical clusters, displaying relationships between search results, and re-clustering results	MeSH, UMLS Semantic Type, lexico-syntactic patterns, TF-IDF, K-Means	No
PubFocus [54]	2006	sorting by impact factor and volume of citations	citations' prioritization algorithm, National Cancer Institute thesaurus and Mouse Genome Database	No

BabelMeSH [32]	2006	multi-language search interface	UMLS, MeSH, WHO EMRO and UMLF	http://allie.dbcls.jp/
CiteXplore [27]	2006	integrating biological literature and data with EBI's tool	fuzzy search	http://www.ebi.ac.uk/citexplore
Hubmed [28]	2006	labeling, saving, and sharing of articles, as well as exporting of citations in many formats	dynamic force-directed graph	http://www.hubmed.org
EBIMED [56]	2007	extracting proteins, GO annotations, drugs and species	co-occurrence analysis	No
eTBLAST [42]	2007	finding documents similar to input text	vector-based TSS algorithms, TF-IDF	No
PubGet [29]	2007	retrieving results in PDFs		No
BioText [35]	2007	searching Open Access Journal abstracts and figure captions, retrieving figures and their associated content	Lucene open source search engine	https://biosearch.berkeley.edu/
Relemed [63]	2007	increasing search specificity	sentence-level co-occurrence	No
EBIMed [56]	2007	displaying proteins, GO annotations, drugs and species	GO	No
PURE [78]	2007	utilizing the user's feedback on the provided papers	Probabilistic model	No
MedEvi [39]	2008	providing ten concept variables of significant biological entities, search results are connected to biological entities and provide citations with matching keywords	search techniques: concordance, positional restriction, semantic restriction and keyword lookup	No
Anne O'Tate [64]	2008	clustering by different aspects including word, topic, journal, author, etc.	TopMine algorithm, UMLS	http://arrowsmith.psych.uic.edu/cgi-bin/arrowsmith_uic/AnneOTate.cgi

MedlineRanker [30]	2009	automatically learning a list of the most discriminating terms for that topic, score, and rank newly published articles	Naïve Bayes	`http://cbdm.mdc-berlin.de/tools/medlineranker`
HONQA [19]	2009	provide question answering system in French and English that can detect the question's model	Naïve Bayes, SVM	No
MiSearch [65]	2009	utilizing implicit feedback for ranking improvement	Ranking algorithm	No
Quertle [18]	2009	performing search with concept categories		No
LigerCat [58]	2009	merging their MeSH descriptions and displaying them in a frequency-based cloud	MeSH, GENBANK	No
RefMed [79]	2010	requesting explicit user comments on relevant documents after getting search results and performing post-learning	RankSVM, DBMS	No
EMERSE [61]	2010	medical record search engine	UMHS	No
iPubMed [71]	2010	instantaneous feedback as the query is typed, enabling approximation search	Interactive and fuzzy searching algorithms	No
AskHERMES [14]	2011	clinical question answering system and outputting question-specific extracting summaries as answers	SVM, lexical tool MMTx, BM25	No
Allie [76]	2011	a repository and search engine for acronyms and their extended forms	UMLS SPECIALIST Lexicon, GENA	No
BioContext [33]	2012	entity recognition, biomolecular event extraction, and contextualization are performed by a number of tools that are extracted, expanded, and combined	GeneTUKit, GNAT, LINNAEUS, McClosky–Charniak, Enju and Gdep parser, TEES and EventMine, GETM	No

GeneView [66]	2012	searching for entities by database identifiers or rate texts according to the number of particular mentions they include	GNAT, MutationFinder, LINNAEUS, ChemSpot, Brno histone modification nomenclature, Alibaba, SVM	No
bioMine [9]	2016	integrating cross-references from numerous biological datasets into a graph model with multiple edge types, such as protein interactions, gene-disease connections, and GO annotations	UMLS	No
BeCAS [51]	2013	selecting several concept types for annotation, accessing other databases, and automatically annotating nested and intercepted concepts	GDep, deterministic finite automatons for dictionary matching; UMLS, LexEBI, Jochem and NCBI BioSystems	No
SciReader [21]	2018	A personalized cloud-based recommender system that identifies articles of interest	LDA, LSA, bag-of-words and KNN methods	No
SemBioNLQA [59]	2019	biomedical quality assurance system with question categorization, document retrieval, passage retrieval, and response extraction modules	lexico-syntactic patterns, SVM, BM25, UMLS, BioPortal synonyms	No
PubMedQA [38]	2019	biomedical QA dataset for answering questions using yes/no/maybe	Fine-tuned BioBERT	No
MedLinker [43]	2020	medical entity recognition and linking	Neural Language Models (NLMs), UMLS	No
TextMed [10]	2020	multiple software agents running on a distributed platform perform the scanning and analysis of biomedical research papers pertaining to the most prevalent ailments in the United States	Reinforcement Learning, Text Mining, MeSH	No

EILEEN [8]	2021	a mechanism for recommending scientific articles and funds	LSA, TF-IDF	No

Numerous systems utilized Unified Medical Language System (UMLS) database for medical entities matching. UMLS is a software with collection of files that consolidates numerous medical vocabularies and standards for pharmaceuticals, disorders, procedures, lab tests, medical devices, creatures, anatomy, and genes [46] published and maintained by NLM. Meanwhile, MetaMap is a highly configurable tool created by Dr. Alan (Lan) Aronson at the NLM to relate medical entities with their UMLS Metathesaurus and, alternatively, to identify Metathesaurus topics mentioned in the text [52]. SemRep is a software that also based on UMLS to extracts three-part propositions from biomedical text phrases [48]. Both applications are based on UMLS.

As can be observed from the list, the evolution of literature recommender systems from the 1980s to the present day has supported a variety of medical text mining. From suggesting medical papers based on interest to taking natural language as query and returning accurate answers to inquiries, the covered subjects range from genes, illnesses to medicine in general. Nevertheless, depending on the specific requirements of the thesis, four systems are studied in further depth. Table 2.2 provides an overview of the four systems, including their main characteristics with pros and cons.

askMEDLINE [31]: Fontelo et al. construct a search engine that allow clinicians, researchers, and average users to identify relevant MEDLINE/PubMed citations by entering natural language query. askMEDLINE does not require domain knowledge to make efficient MEDLINE/PubMed searching.

The application employs a multiple search rounds strategy. In the initial phase, the parser disregards punctuation and eliminates terms from a "stop-word" list. The amended query is rephrased by a parser and matched with various keywords based on its relevancy to biomedical terms.

In the first round, the terms that match MeSH terms, a medical vocabulary thesaurus published by NLM, are selected. Those terms are sent back to PubMed to retrieve results. If the number of articles retrieval is in the range of 1 to 50,000, 20 top results are presented and the search procedure ends. If no journals are discovered or the number of retrieved articles exceeds 50,000, the search may continue to Round 2. Depending on whether the retrieval count is too low or too high, terms will be added or omitted from the query. Similarly, if the count of articles received from the second round falls between 1 and 50,000, the search procedure ends. If not, the search will continue by adding or eliminating relevant biomedical terms. A result between 1 and 50,000 citations finishes the procedure and presents the first 20 results.

Developed by a team of NCBI professionals, askMEDLINE utilizes most of this service's resources. askMEDLINE depends solely on PubMed's MeSH vocabulary for biological entity recognition and Entrez E-Utilities for articles retrieval. In addition, the multiple-rounds search strategy to automatically retrieve keywords makes this method simple but possibly less effective than manual inputs. We observed that MeSH vocabulary thesaurus can not compete with our medical researchers in enriching list of keywords. Moreover, the application does not employ any text mining techniques to comprehend questions or further explore provided abstracts, therefore its main task is sourcing relevant keywords. A web API is maintained to aid users in their searches, and it is still accessible as of the writing of this thesis.

	askMEDLINE [31]	**AskHERMES [14]**	**SemBioNLQA [59]**	**BioMine [9]**
Key features	return relevant journal articles	integrate info from different sources to formulate answer	retrieve relevant passages & extract answer	retrieve relevant passages & extract answer
Data source	PubMed abstracts	PubMed, eMedicine, Wikipedia & clinical guides	PubMed abstracts	PubMed abstracts & fulltexts
Answer type	list of articles	exrtractive summary	short precise & ideal answer	list of articles
Method	multi-round search with predefined rules	supervised ML, BM25, clustering	ML, BM25, NLP	search platform Solr
Pros	take good advantage of PubMed resources	end-to-end service	end-to-end service	simple to build, support extensive features
Cons	simple approach	require powerful server (six servers)	fit for BioASQ challenge	hard to modify, inefficient way to import data

Table 2.2: In-depth comparisons of medicine research paper recommender systems

AskHERMES [14]: Figure 2.1 depicts the architecture of the system, which accepts natural language query as input. First, Question Analysis extracts information automatically from the query and generates a list of query phrases. UMLS database is used by the system to expand search terms. Similar questions are returned using the Related Questions Extraction module. Locally indexed documents are returned by the Information Retrieval function. The main function of Information Extraction is retrieving most relevant passages. Lastly, those relevant passages are gathered and summarized by Summarization & Answer Presentation.

AskHERMES collected data from various literature sources in medicine fields including MEDLINE, eMedicine, PMC and Wikipedia, and then manually curated filters were applied to maintain semantic information. For each table, all textual data is tied with its header and caption into a separate passage, and all section titles would be combined with any sentences within the respective section. Using supervised machine-learning techniques, a question is automatically classified based on general topic and its keywords are identified automatically. While SVM is used to classify questions, Conditional Random Field (CRF) is used to identify keywords, and MetaMap is applied to map UMLS concepts and semantic categories. AskHERMES then incorporated the most recent version of the probabilistic relevance model BM25 for document retrieval. BM25 is a document ranking model that use bag-of-words method to rank document with relevancy to search query [5]. Passage retrieval is accomplished by implementing a document scoring function, considering the score of document similarity with respect to the question, the frequency of search term that appears in the sentence, the number of unique search terms in the sentence and the similarity between the sentence with the question.

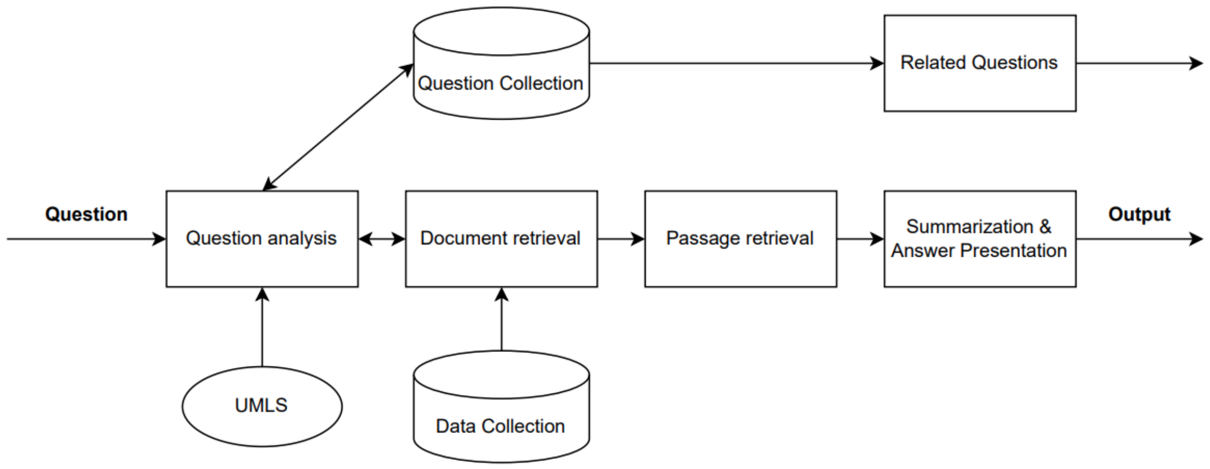


Figure 2.1: The AskHERMES architecture [14]

AskHERMES is distributed on six servers using the Linux/Solaris operating system. Its average system response time is 20 seconds. The web application, however, is not updated and therefore currently inaccessible. Overall, it is a comprehensive system that provides a whole range of services, from processing natural language questions to summarizing answers using many ML techniques. However, the program's operation requires a strong server, which is not always feasible for small businesses.

SemBioNLQA [59]: Sarrouti et al. presented SemBioNLQA, a question-answering system that take question as input and return exact as well as ideal responses to the question. Using different techniques from ML algorithms, BM25 model to UMLS and BioPortal database, the system offers an end-to-end service that is illustrated in Figure 2.2.

Each biomedical question that is classified by SemBioNLQA's question classification function will be divided into one of the following types: yes/no, factoid, list, or summary. Those categories are defined by one of BioASQ's challenges. Using constructed lexico-syntactic patterns, the computer trained the classifier using 810 BioASQ [1] training questions. The question is initially represented as a vector of phrases. Then, a ML system categorizes the given query into one of the four previously described groups. MetaMap is used to map the terms of the question to the UMLS meta thesaurus to retrieve medical identities for the document retrieval query. Document retrieval is done using E-Utilities to extract document from PubMed.

For document retrieval, a generated query is sent to PubMed using the E-utilities to get relevant MEDLINE articles. The returned papers are re-ranked using UMLS database, which computes the relevance score between biological topics in the question and the title of all returned articles. Depending on the question type, several tactics and tools, such as SENTIWORDNET, SENTIWORDNET, BM25, UMLS, and TF-IDF will be used to offer answers.

Overall, SemBioNLQA provides a vast number of services, particularly in the field of answer summarization. However, the program was limited to four question types stated for the BioASQ challenge, as it was designed to participate in the challenge. The system's source code is publicly available for further investigation and update.

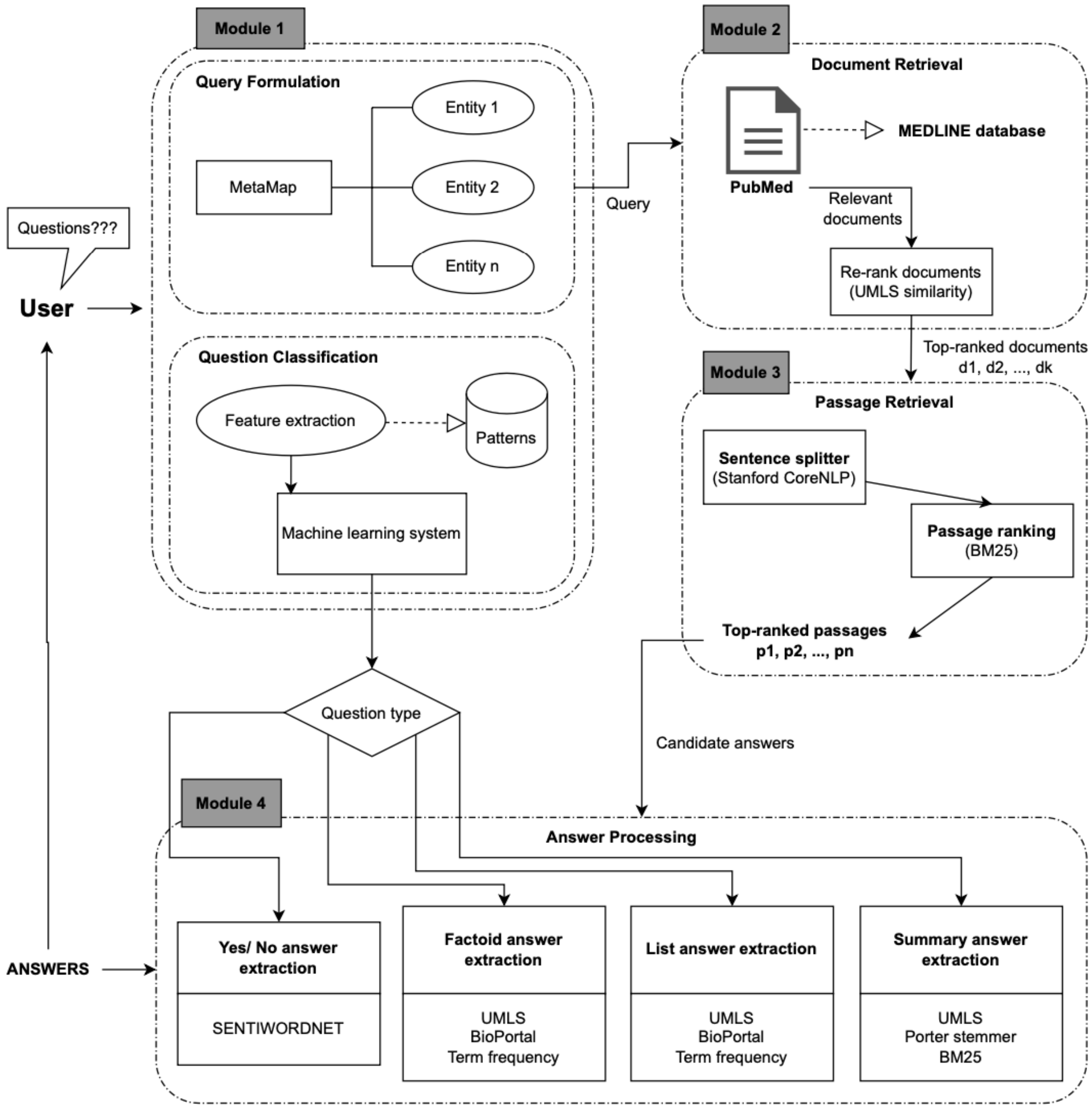


Figure 2.2: Overall architecture of SemBioNLQA [59]

BioMine [9]: BioMine is a search engine for full-text medical research papers that accept natural language query introduced by Almeida et al. in their 2016 publication. The system transforms the search queries from their natural language format then augments with medical terminology from the UMLS Metathesaurus.

BioMine was created utilizing databases from PubMed in XML format and PMC in NXML format. The indexation module of bioMine is based on the search engine Solr that is open-sourced. Solr is an enterprise-ready, lightning-fast, and extremely scalable search engine that uses Apache Lucene that is ready for the cloud and operates as a standalone server [62]. A parser is developed to process all XML and NXML files and populate the index fields used by Solr to index documents. The complex query module processes queries given by users. In addition, bioMine queries are categorized based on the user's perceived needs and processed using various ways to enhance the result's relevancy.

Due to the fact that the system was built on Solr, it does not require any further methods besides UMLS to retrieve relevant articles. Therefore, the primary contribution of the system is not its approach, but rather its implementation in real-world situations. On one hand, it makes the system easier to construct and more stable, but on the other hand, provides fewer options for modification. Additionally, in order to extract articles, Solr demands that databases be uploaded to their servers, which makes the database import procedure inefficient and necessitates large amount of work to maintain and update the database. The system's source code is openly available for implementation and reconstruction by academics.

2.2 BM25

BM25, or Okapi BM25 in its full form, is a ranking function for calculating documents' relevancy compared to a particular search query [57]. The name Okapi BM25 comes from the fact that the function is first used in Okapi search engine in the 1980s and 1990s [5]. BM25, which is an abbreviation for "Best Match 25", began to be utilized by an increasing number of researchers during TREC tournaments [68]. According to Trotman and Keeler, BM25 performs close to human levels on the TREC datasets [67].

BM25 relies on bags-of-word model to ranks documents according to the query's keywords that exist in each document, regardless of the keyword's order within the text. The function assigns a score for the degree to which a query with terms $q_1, ..., q_n$ matches an indexed string field t in a document D. The score is computed as [2]:

$$\sum_{i}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{field\_len}{avg\_field\_len}\right)}$$

Where the function's components are:

$f(q_i, D)$: The number of occurrences of query term i in field t of document D (term frequency). For multi-value fields, the total of all element occurrences is used.

$field\_len$: The field length (in number of words) of document D's field t. For multi-value fields, the sum of field lengths across all elements is utilized.

avg_field_len: The average field length of field t across all content node documents.

k_1: A parameter that limits the impact of a single query phrase on document D's score. With a greater value, the score for a single term can continue to increase compared to the number of instances of that term. The default value is 1.2. Configurable through rank properties.

b: A parameter that controls the influence of field t's field length relative to the average field length. The default value is 0.75. Configurable through rank properties.

$IDF(q_i)$: The Inverse Document Frequency (IDF) of the search phrase i within the field t. This is computed as:

$$log\left(1 + \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}\right)$$

where N represents the total number of documents included on the content node. $n(q_i)$ is the number of documents that contain query term i for field t, calculated for each index that exists for that field. As the IDF is calculated per content node and index, some deviations are possible. To utilize the same IDF across all content nodes, its importance on each query word through annotations is set.

BM25 is overall a text-only ranking function that operates on an indexed string field. The function is inexpensive to compute while it can yield a high-quality rank score. Hence, its usages are numerous. BM25 is typically used as a comparative standard for other functions. In addition, BM25 and its variants can be employed as a standalone ranking function or as a first phase ranking in a pipeline that can be re-ranked by various learning to rank algorithms [68]. Some of the most popular search engines, including Okapi, Lucence, and ElasticSearch, utilize BM25 as their default scoring function. AskHERMES and SemBioNLQA are two candidates in the field of recommender systems for medical research papers that utilized BM25 as part of their pipelines.

2.3 BERT

BERT stands for Bidirectional Encoder Representation from Transformer, an embedding model that was unveiled by Google in 2018 [22]. Since its introduction, it has generated a revolution in the field of NLP by producing improved outcomes in a variety of NLP tasks, including named-entity recognition, question-answering, sentence classification, etc. The fact that BERT is a context-based embedding model, unlike other models such as word2vec, is one of the primary reasons for its popularity [55]. First, the thesis provides an overview of the transformer architecture, then BERT's architecture and training procedure are discussed. The following presentation of the transformer architecture in general and BERT in particular are based on the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding" [22] and the book "Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT" [55].

2.3.1 Transformer

Given that BERT is a transformer-based model, it is essential to understand the notion of transformer in order to comprehend BERT's architecture.

Transformer is a novel neural architecture that encodes the input data as potent features using the attention mechanism described in "Attention Is All You Need" [70]. Unlike traditional architectures, it solely relies on attention mechanisms, eliminating the need for recurrence or convolutions. The main components of the transformer include encoder and decoder. The encoder takes the source sentence as input, learning its representation and passing it to the decoder. The decoder, utilizing the encoder's acquired representation, generates the target sentence as output. The general architecture of a transformer is depicted in Figure 2.3. The key components of a transformer include:

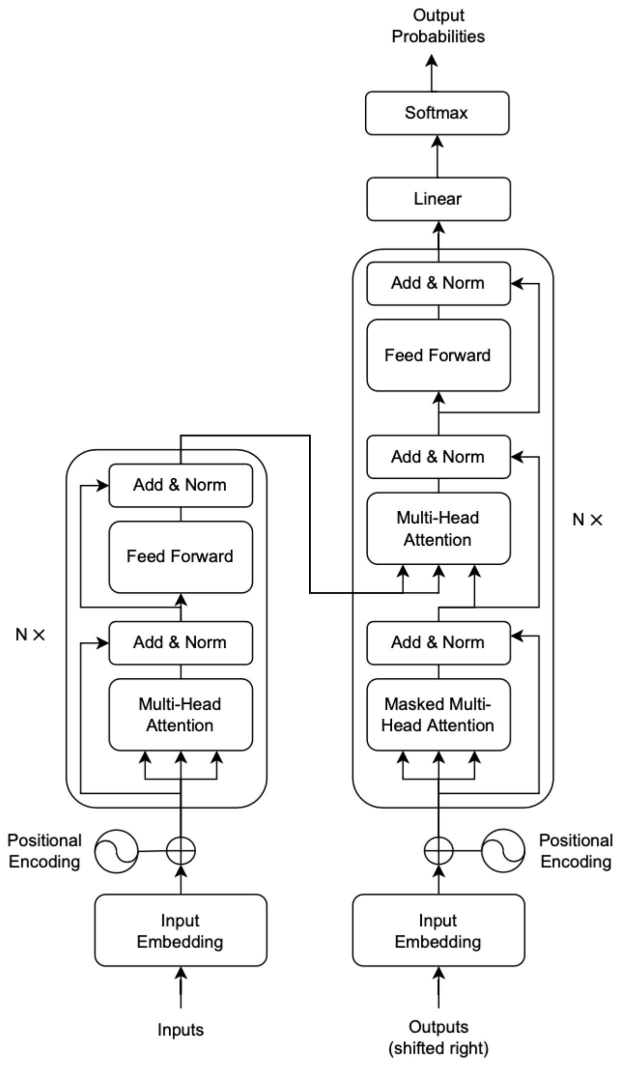


Figure 2.3: The Transformer - model architecture [70]

Input Embedding: the input sequence is first mapped with the transformer into a vector space that is continuous. Using an embedding layer, each token is mapped to a compact vector representation.

Positional Encoding: as the transformer lacks a sequential structure, it requires a method to account for the position of tokens in the sequence. To accomplish this, a positional encoding vector is added to the embedded tokens.

Multi-Head Attention: at the core of the transformer is the multi-head attention mechanism. This mechanism allows the model to focus on different segments of the input sequence, thereby efficiently capturing long-range dependencies. Multi-head attention is achieved by calculating the dot product between a query and a set of key-value pairs. The weighted result is then combined using a softmax function to form the output.

Feedforward Layer: following each multi-head attention layer, a feedforward layer applies a point-wise fully connected network to each position independently and in the same manner.

Decoder-Encoder Architecture: in the original transformer paper, the model was applied to sequence-to-sequence tasks, such as machine translation. There are two sets of multi-head attention layers in this instance: one for the encoder and one for the decoder. Encoders process input sequences, while decoders generate output sequences. During training, the output sequence is provided with a time step delay as input to the decoder.

2.3.2 Model architecture

BERT has the architecture as a multi-layer, bidirectional Transformer encoder based on the implementation provided in the paper "Attention is all you need" [70]. The encoder of the transformer is called bidirectional as it can read a sentence in both directions. The transformer recognizes the context of each sentence and provides contextual representation of those sentences as output.

In BERT, L represents the layers' number, H represents the hidden size, and A represents the self-attention heads' number. Different BERT configurations are developed based on the number of layers and hidden size.

The BERT-Base and BERT-Large configurations are two standard options. BERT-Base includes twelve layers (L=12), each layer includes twelve attention heads and the network has a hidden size of 768 (H=768) in total. There are 110 million parameters in the BERT-Base. In comparison, BERT-Large includes 24 encoder layers, each layer has a number of attention heads of 16 and the network has 1024 hidden units in total. Thus BERT-Large has 340 million parameters [55]. In addition to the preceding two conventional configurations, the BERT model can be constructed with other combinations. These configurations are shown in the table 2.3.

	H=128	**H=256**	**H=512**	**H=768**
L=2	2/128 (BERT-Tiny)	2/256	2/512	2/768
L=4	4/128	4/256 (BERT-Mini)	4/512 (BERT-Small)	4/768
L=6	6/128	6/256	6/512	6/768
L=8	8/128	8/256	8/512 (BERT-Medium)	8/768
L=10	10/128	10/256	10/512	10/768
L=12	12/128	12/256	12/512	12/768 (BERT-Base)

Table 2.3: 24 BERT configurations [69]

In environments with low processing resources, BERT can be implemented with smaller configurations such as BERT-Tiny, BERT-Mini, BERT-Small or BERT-Medium. Nevertheless, BERT-Base and BERT-Large, the typical BERT configurations, can provide more accurate findings and thus are more widely used.

Before feeding the input to BERT, first the model will split sentences into tokens using basic tokenizer, WordPiece tokenizer [74] and special tokenizer. Basic tokenizer is used to separate words with punctuation marks while keeping special characters like conjunctions "&" or "-". After the text is splitted by basic tokenizer, WordPiece tokenizer is used to further generate tokens for embedding. If a word belongs to a pre-defined dictionary of 30K tokens, it is considered a token. Otherwise, the word is separated into subwords and compared those 30K tokens until all individual characters are splitted. Consider an example of two sentences: "Python is easy to learn. I like learning Python." In the example, WordPiece tokenizer's dictionary does not include the word "learning". Therefore, the word "learning" is divided into its component parts, learn and ##ing. The hash marks preceding the character ##ing show that it is a subword preceded by other words. WordPiece tokenizer checks to see if the subword ##ing exists in the vocabulary. As it already exists in the vocabulary, it is not separated and utilized as a token. Other tokenizers which are used include tokens [SEP], [CLS] and [MASK]. While [CLS] is used to mark the start of the first sentence, [SEP] is used to mark the end of each sentence. [MASK] token marks the position of masked words in Masked language modeling task for pre-training that will be discussed later.

Then, the input is converted into embeddings with three layers: token embedding, segment embedding, and position embedding. The function of token embedding is to translate all tokens, such as $E_{[CLS]}$ for [CLS] token embedding and E_{Python} for Python token embedding. To distinguish between these segments, BERT adds segment embeddings to the input tokens. Lastly, position embedding adds positional information to the token embeddings to capture the order of the tokens within the input sequence. Figure 2.4 represents a graphic representation of the three embedding layers structure.

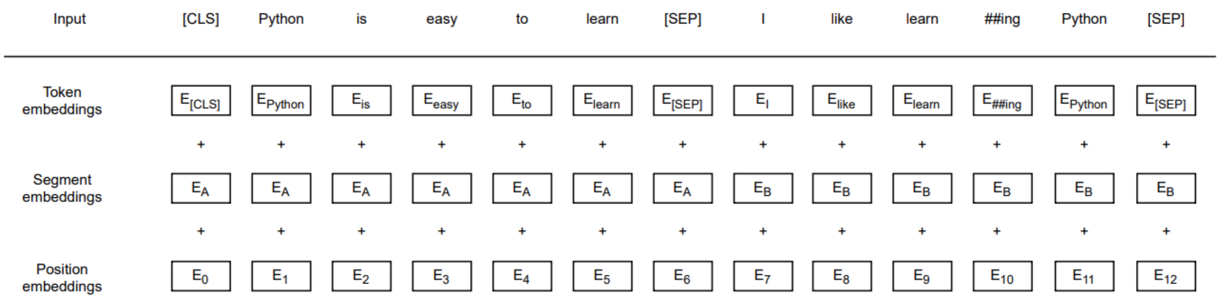

Input
[CLS]
Python
is
easy
to
learn
[SEP]
I
like
learn
##ing
Python
[SEP]
Token embeddings
$E_{[CLS]}$ E_{Python} E_{is} E_{easy} E_{to} E_{learn} $E_{[SEP]}$ E_{I} E_{like} E_{learn} $E_{\#\#ing}$ E_{Python} $E_{[SEP]}$
Segment embeddings
E_A E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B E_B
Position embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10} E_{11} E_{12}

Figure 2.4: Embedding presentation of the input [22]

The building of BERT comprises of two main stages: pre-training and fine-tuning. First, the model is trained with unlabeled data across a variety of pre-training tasks include Masked language modeling and Next Sentence Prediction. Initialized with the pre-trained parameters, the BERT model is next fine-tuned using data with labels from downstream tasks. Even though all tasks initiated with the same parameters from pre-training, each downstream task has its own customized model. Figure 2.5 demonstrates BERT's framework with three fine-tuning tasks including SQuAD, NER and MNLI.

2.3.3 Pre-training

The pre-training phase of the BERT model involves two primary tasks: Masked language modeling (MLM) and Next Sentence Prediction (NSP).

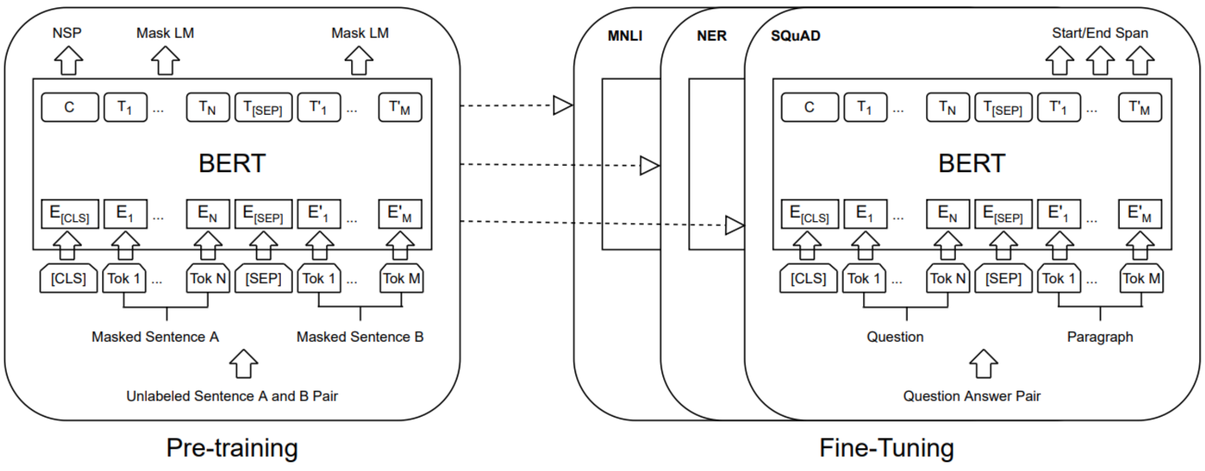


Figure 2.5: Overall pre-training and fine-tuning procedures of BERT [22]

Masked language modeling to make predictions, BERT reads the text in both ways: right-to-left and left-to-right. To perform MLM, 15% of the words in a given input sentence are randomly masked. The model attempts to anticipate the masked words using bi-directional techniques that read sentence in both ways.

Consider the same example: "Python is easy to learn, I like learning Python." Initially, the sentences are tokenized to obtain the tokens:

$$tokens = [[CLS], Python, is, easy, to, learn, [SEP], I, like, learn, \#\#ing, Python, [SEP]]$$

Next, 15% of the tokens are obscured at random.

$$tokens = [[CLS], Python, is, easy, to, [MASK], [SEP], I, like, learn, \#\#ing, Python, [SEP]]$$

Now, the BERT model attempts to predict the tokens that are masked. To forecast the masked token, BERT uses a feedforward network along with softmax activation. As shown in Figure 2.6, the feedforward network returns the probability that each word in the lexicon is the masked word. According to the following graphic, there is a good likelihood that the disguised word is "learn." In this situation, "learn" will be predicted as the masked word.

Another method of masking input tokens is whole-word masking. Continuing with the same scenario, here are the masked tokens:

$$tokens = [[CLS], Python, is, [MASK], to, learn, [SEP], I, like, learn, [MASK], Python, [SEP]]$$

As may be seen in the previous tokens example, the terms "easy" and "##ing" have been masked. Nota bene: the word "##ing" is a subset of the word "learning". In the process of masking the entire word,

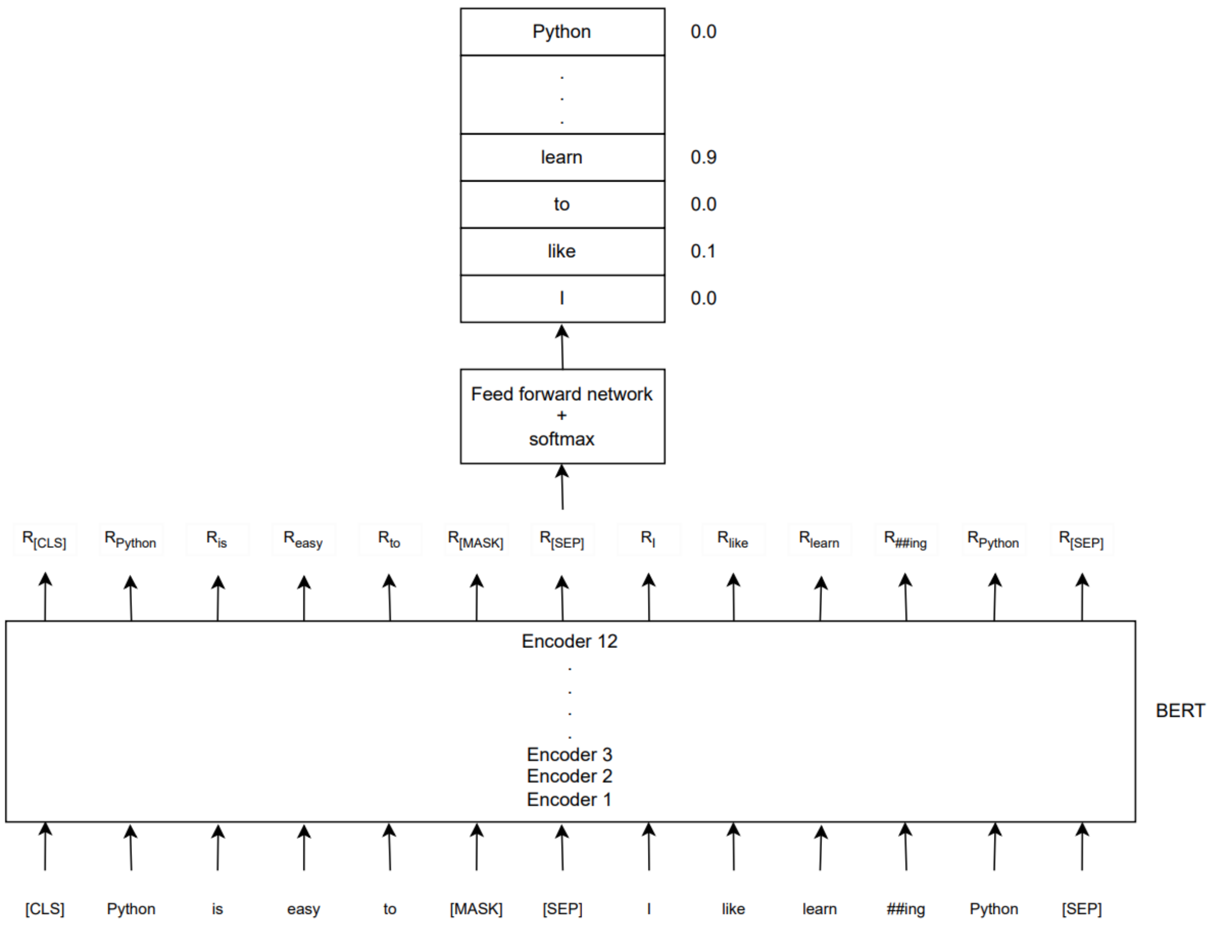

Python 0.0
learn 0.9
to 0.0
like 0.1
I 0.0
Feed forward network + softmax
R[CLS] RPython Ris Reasy Rto R[MASK] R[SEP] RI Rlike Rlearn R##ing RPython R[SEP]
Encoder 12
Encoder 3
Encoder 2
Encoder 1
BERT
[CLS] Python is easy to [MASK] [SEP] I like learn ##ing Python [SEP]

Figure 2.6: Predicting the masked token [55]

if the subword is masked, all the words that belong to it are also masked. Therefore, the tokens are now the following:

$$tokens = [[CLS], Python, is, [MASK], to, learn, [SEP], I, like, [MASK], [MASK], Python, [SEP]]$$

Next Sentence Prediction (NSP) helps the model understand the relationship of two consecutive sentences in a text. NSP is a binary classification that determines whether a sentence is the next sentence of another sentence or a randomly selected sentence from the corpus. Consider two examples below:

A: Today is a sunny day. B: I would like to go outside. (1)

A: Today is a sunny day. Sentence B: The food is delicious. (2)

Sentence B in the first example is the actual follow-up sentence of sentence A, so sentence B is labeled as *isNext* while sentence B in the second example is labeled *notNext*. Note that 50% of data points must be maintained in the *isNext* class and 50% in the *notNext* class.

Pre-training procedure The data sources for pre-training BERT are BooksCorpus (800 million words) and English Wikipedia (2,500 million words). It enables the model to comprehend the English language in the general domain. Only text excerpts are exclusively taken from Wikipedia and lists, tables, and headings are disregarded. After tokenization, the tokens are supplied to BERT in order to guess the masked tokens or whether a sentence is the next sentence of another one.

BERT is trained with a batch size of 256 sequences plus 512 tokens to 128,000 tokens/batch for one million steps. It is run for 40 epochs on the 3.3 billion word corpus. The paremeter as Adam optimizer [40] is: $lr = 1e-4$ learning rate, $\beta_1 = 0.90$, $\beta_1 = 0.999$, L2 weight decay = 0.01, learning rate warm up is 10,000 steps, and using linear learning rate decay. Notably, the learning rate is a hyper-parameter that controls the rate at which an algorithm updates or learns the estimated values of a parameter. It shows how frequently the neural network updates its learned notions. Smaller learning rates demand more training epochs because changes are slowly made, but it might be easier to converge on something meaningful. The warm-up phase consists of 10,000 iterations during which the learning rate gradually increases from 0 to 1e-4 in a linear fashion. After 10,000 iterations, learning rate is decreased linearly as convergence approaches. Also, depending on whether the text was lowercased or not prior to the WordPiece tokenization stage, two versions of each BERT model are created: uncased and cased. In general, the BERT-uncased model is utilized more frequently; but, where the cases must be kept, as in Named Entity Recognition (NER) task, BERT-cased has proven to be more effective.

With those setting parameters, BERT-Base was trained on 16 TPU chips in total. Meanwhile, BERT-Large pretraining was executed on a total of 64 TPU chips. Each pretraining session lasted four days. Pretraining is known to be costly, but it is a one-time process for each language, after which extracted weights can be applied to various downstream tasks. Therefore, the majority of researchers are not required to train their own models from scratch.

2.3.4 Fine-tuning

After BERT has been "taught" the language through pretraining, the model must be fine-tuned to handle downstream tasks. Fine-tuning entails that BERT is not trained from scratch; rather, pre-trained weights are utilized and adjusted to feed each task. In general, fine-tuned BERT models are produced by adding one additional layer after the final BERT layer and training the complete network for only a few epochs.

The parameters for fine-tuning are mostly the same with pre-training except for learning rate, batch size and the amount of training epochs. The probability of dropout was constantly maintained at 0.1. The best hyperparameter values vary for each task, however, the following range is applicable [22]:

- **Batch size**: 16, 32.
- **Learning rate (Adam)**: 5e-5, 3e-5, 2e-5.
- **Number of epochs**: 2, 3, 4.

BERT has been shown to outperform other models in eleven NLP tasks, including sentiment analysis, sentence classification, question-answering, text summarization, etc [22]. The following part will focus on techniques for fine-tuning passage ranking/re-ranking.

Passage ranking/re-ranking Nogueira et al. [50] proposed a fine tuned BERT model for query-based passage re-ranking in 2019. A question-answer pipeline comprises three primary stages. First, a standard algorithm, such as BM25, retrieves from the corpus a large number (for instance, 1,000) of potentially relevant documents to a given query. Next, relevance score of those passages are calculated and re-ranked using an approach that is more computationally costly. The top ten or 50 of these texts will then be used by an answer generation module as the source for candidate responses. In the paper, the process of this pipeline's second stage, passage re-ranking is discussed.

The re-ranker is responsible for estimating the relevancy between a query to its candidate passage. BERT is utilized as a re-ranker. Using the same nomenclature as Devlin et al. [22], the natural language query and the passage text are fed as input. The query was pruned to contain maximum 64 tokens while the candidate passage should not have more than 512 tokens. BERT-Large and BERT-Base models are employed as binary classification models to calculate the passage's relevance probability [4]. Each passage then will have its own relevance probability, and the list of passages is ranked according to these probabilities. Fine-tuning of BERT for passage re-ranking is done using cross-entropy loss function.

BERT for passage re-ranking is fine-tuned with batch size of 128 plus 512 tokens and 100K iterations, which is run for 30 hours on a TPU v3-8. Fine-tuning BERT for passage re-ranking also employs Adam optimizer of learning rate of $lr = 3x10^-6$, $\beta_1 = 0.90$, $\beta_1 = 0.999$, L2 weight decay = 0.01, and linear learning rate decay. It is noticeable that this fine-tuning work has a slower learning rate than other fine-tuning tasks and BERT pretraining itself. It may result in improved performance and precision, but will be more demanding on the server and so more costly.

2.4 BioBERT

BioBERT is a biomedical language model which is pre-trained using the BERT architecture. Created by researchers at Seoul National University and the Korea Institute of Science and Technology Information, BioBERT is built in order to enhance the performance of NLP for biomedical domain [41]. BioBERT is claimed to beat BERT on three different NLP tasks in biomedicine field. Figure 2.7 summarizes the process of pre-training and fine-tuning BioBERT.

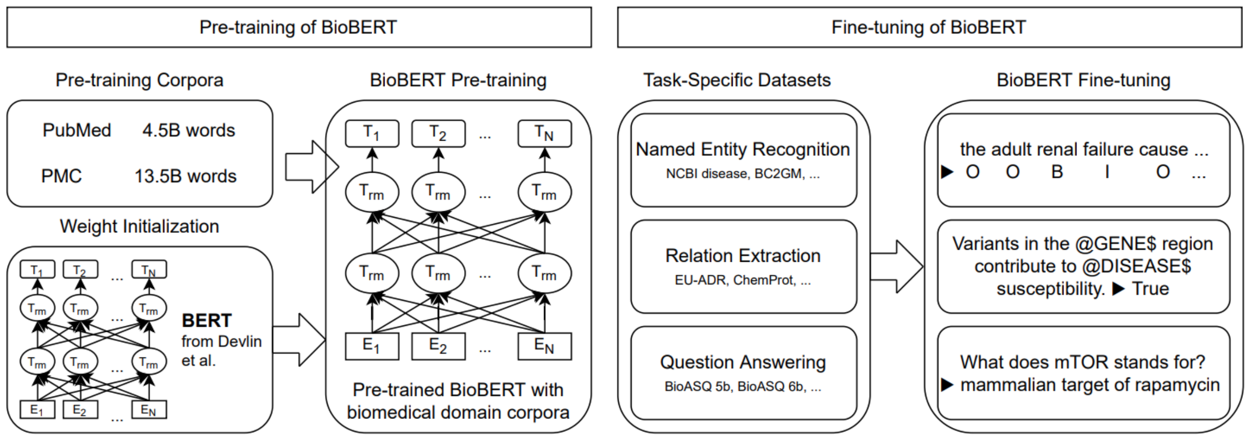


Figure 2.7: Overview of the pre-training and fine-tuning of BioBERT [41]

Pretraining BioBERT is pre-trained for 23 days on biomedical corpora using eight NVIDIA V100 GPUs. It is pre-trained with the help of abstracts from PubMed of 4.5B words and full-text papers from PubMed Central (PMC) database of 13.5B words. As BioBERT uses the initial checkpoint from BERT, BioBERT is also considered to be pre-trained by English Wikipedia and BooksCorpus as with BERT. BioBERT also employs WordPiece tokenization to handle biomedical terms by splitting them into common subwords. For example, the term Leukemia can be written as Leu + ##ke + ##mia. The authors of BioBERT also realized that utilizing capitalized language (rather than lowercase) results in improved performance. Different pre-trained BioBERT weights are available for download depending on the initial BERT weights and text corpus used for pre-training. Two latest BioBERT versions are BioBERT-Base v1.1 and BioBERT-Large v1.1, both are pre-trained on PubMed abstracts.

Fine-tuning after pre-training, BioBERT is fine-tuned for three text mining tasks in biomedical field includes named entity recognition, relation extraction, and question answering.

Named Entity Recognition: In this task, BioBERT is taught to recognize and label named entities in biomedical text, such as genes, proteins, diseases, and drugs. Each token is tagged with its corresponding entity type using the BIO (Beginning, Inside, Outside) labeling scheme on the tokenized input text. Then, BioBERT is trained to predict the appropriate entity type for each token based on its context.

Relation Extraction: In this task, BioBERT is taught to identify and classify the semantic relations between entities in biomedical text. BioBERT is trained to determine, given a sentence that mentions a gene and a disease, whether the gene is associated with the disease. (e.g., "BRCA1 is a risk factor for breast cancer").

Question Answering: In this task, BioBERT is taught to answer queries based on biomedical text in this task. BioBERT is trained to extract the answer to questions such as "What is the function of gene ABC?" by identifying the relevant sentence or paragraph in a biomedical text.

3 Problem definition

Bloom Diagnostics GmbH is a health technology startup with offices in Switzerland, Austria, and Germany, headquartered in Zurich, Switzerland. By providing the Bloom Tests and Bloom Lab which is a blood test along with an electronic test reader that users can have at home, Bloom empowers its customers to take charge of their own health [23]. The Bloom Test is intended to detect certain biomarkers in a few drops of blood. Using known lateral flow immunoassay technology, the test provides blood values for analysis. The Bloom Lab was created for analyzing Bloom Test strips. In conjunction with the user's personal information, the Bloom App then generates a personalized report of the health status of customers based on their blood test results with medical advice via the smartphone app. Figure 3.8 depicts a general perspective of Bloom's system. Bloom presently offers the Bloom Ferritin, Bloom Thyroid, Bloom Inflammation, and Bloom Kidney Tests, and is constantly striving to expand its product line [24].

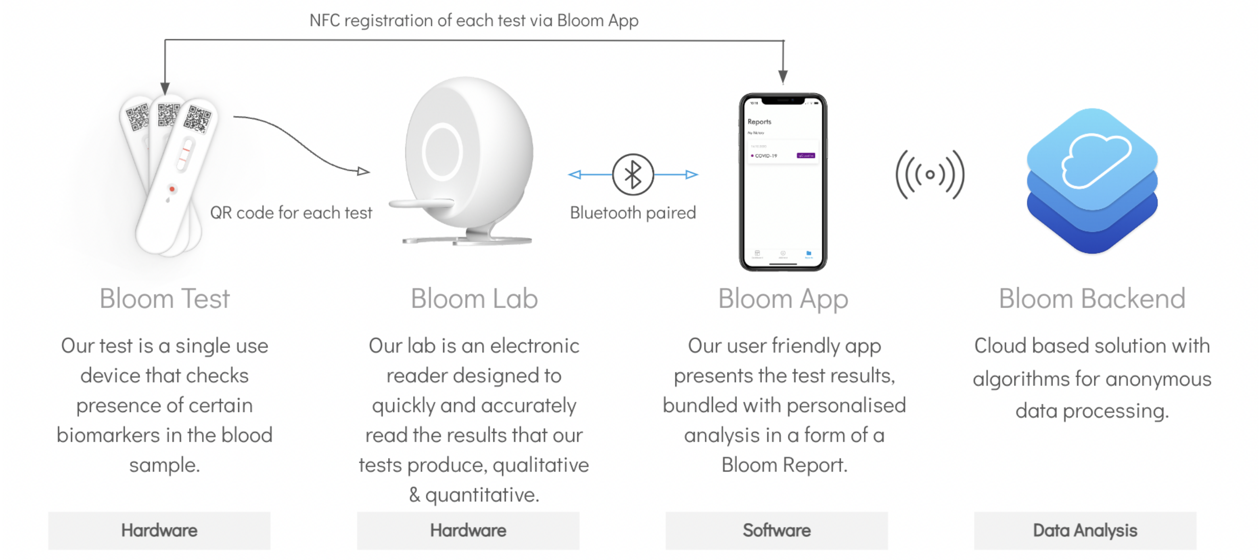
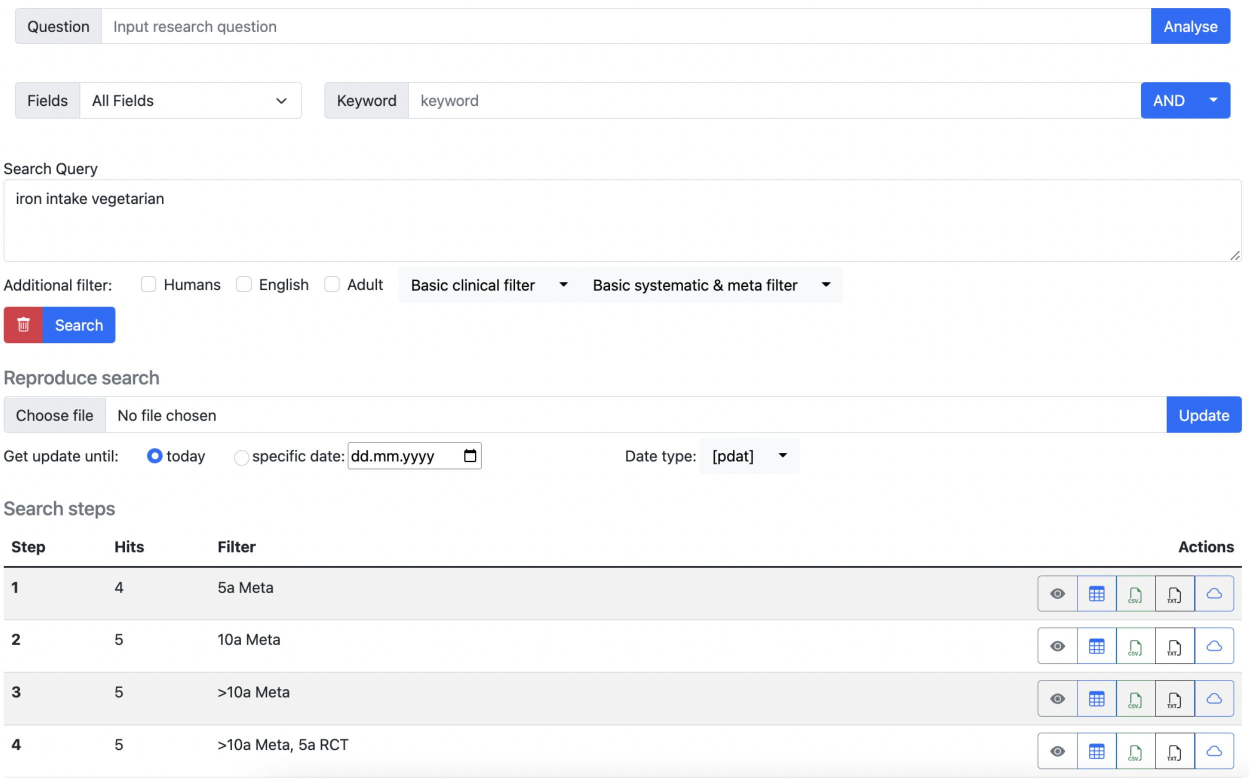

NFC registration of each test via Bloom App
QR code for each test
Bluetooth paired
Bloom Test
Our test is a single use device that checks presence of certain biomarkers in the blood sample.
Hardware
Bloom Lab
Our lab is an electronic reader designed to quickly and accurately read the results that our tests produce, qualitative & quantitative.
Hardware
Bloom App
Our user friendly app presents the test results, bundled with personalised analysis in a form of a Bloom Report.
Software
Bloom Backend
Cloud based solution with algorithms for anonymous data processing.
Data Analysis

Figure 3.8: An overview of Bloom's products system

In order to provide a wide variety of blood tests for different indexes of the body including ferritin, inflammation, kidney or thyroid level with personalized medical advice to a broad range of clients from various origins, ethnicities, and age groups, huge amounts of medical data and literature must be investigated. The medical literature review became one of the most significant tasks that had to be completed by a team of specialized medical researchers, undergoing a literature search and evaluation process. These procedures ensure that there is sufficient clinical evidence to support medical advice for customers and establish the scientific validity and clinical performance of each test.

When a medical team defines a study interest and formulates questions related to blood measurements, for instance which factors can affect the rise of ferritin level in blood, the literature search and evaluation process begins. The search will be conducted in multiple steps according to study types and publication dates. The nature of the search is time-consuming, repetitive, and requires a high level of expertise, making it costly to perform.

PubMed is one of the medical team's most reliable medical research papers platform. A web application has been developed by the company based on PubMed in order to support the company's scientific research efforts. The architecture of the software consists of a server backend running on AWS that communicates with PubMed, and a web page frontend for entering queries and retrieving results. All searches are conducted on PubMed, and commands are supplied over its REST API. This online application is capable of performing multiple functions, including keywords recommendation, systematic search execution, metadata and abstract extraction, as well as word cloud generation from abstracts. By connecting the SemRep and MetaMap databases which contain medical terminologies to the natural language questions, a list of suggested keywords can be generated. Utilizing SemRep and MetaMap, the application maximizes the number of extracted biomedical-related terms. Researchers can also add their own keywords based on the terms extracted by the application. Systematic search is the most essential function of the online application. Based on the company's search and evaluation process, the application performs a search with the corresponding search query and scrapes metadata and abstracts using the public APIs that the NLM provides. Depending on their study objectives, medical teams may employ various filters of different studies and articles published time to receive a list of articles with metadata and abstract. With these abstracts, the computer may generate a word cloud with varying collocation thresholds and minimum word lengths in order to provide researchers with an overall view of the papers discovered. Figure 3.9 displays the application's user interface.



Search Query
iron intake vegetarian

Additional filter: Humans English Adult | Basic clinical filter | Basic systematic & meta filter

Get update until: [✓ today] specific date: dd.mm.yyyy Date type: [pdat]

Search steps

Step	Hits	Filter	Actions
1	4	5a Meta	
2	5	10a Meta	
3	5	>10a Meta	
4	5	>10a Meta, 5a RCT	

Figure 3.9: An overview of Bloom's Research Web Application

Although the application can already assist medical researchers with certain tasks, after obtaining a list of publications, researchers must still browse through each document to extract meaningful information. This process may need to be done numerous times with different keywords, which can significantly increase the amount of time required to investigate documents to acquire knowledge. The time-consuming process of the medical team inspires us to investigate natural language methods that can further assist researchers in their literature search & review process. The medical team needs a research paper recommender system that can rank retrieved articles in a relevant way to help them save time analyzing every document. The research paper recommender system might have the following characteristics:

First, the system should be specialized for the medical field but not limited to specific areas, such as genes or molecular. As the scope of the company's products may extend in the future and researchers may need to broaden their research areas, the system will need to address research interests for all subtopics in medicine, such as infectious diseases, genes, or immunology.

Secondly, the system might accept natural language questions and queries as input. After receiving questions and queries, the system analyzes them to extract keywords and uses those keywords to send to PubMed for a list of potentially relevant articles. The passages within those articles are compared with the original natural questions and queries to find the most relevant passages to the question or query.

Last but not least, a feasible solution for saving time for medical researchers is for the system to yield an ordered list of relevant articles. As researchers are required to support their claims with proof and citations, a ranked list of publications is preferable to a summarized answer so that they can verify and add citations to their conclusions. Ideally, the recommendation should likewise rank all paragraphs inside the articles in accordance with question relevancy. This feature will reduce research time by highlighting possibly relevant passages to the question first. Currently, PubMed also has the function with the option "Sorted by: Best match" to display the most relevant passages and documents. However, those rankings are mostly based on abstracts or titles without considering full texts.

In conclusion, motivated by the current situation at Bloom Diagnostics and the research activities of the medical research team, this thesis aims to develop a research paper recommender system in medicine that is customized for Bloom and complements their existing research application. The system must be compatible with Bloom's research efforts, hence it must be able to assist in answering numerous types of medical questions without being limited to a certain field. After assessing the question and related articles, the system can offer the most relevant passages and, based on these passages, relevant articles.

4 Approach

A general process of building a recommender system for research papers is described in this chapter. Figure 4.10 describes the generic process of the main functions of the recommender. Overall, there are four main steps followed by the recommender system for research papers: Question pre-processing, Document retrieval, Text pre-processing, and Passage ranking. The overall process is generalized from three systems that are described closely in the Chapter 2 include AskHERMES, askMEDLINE, SemBioNLQA. The concept also takes into account the company's present research process and ensures that the recommender is easy to maintain and does not require large storage space that can be built at Bloom Diagnostics.

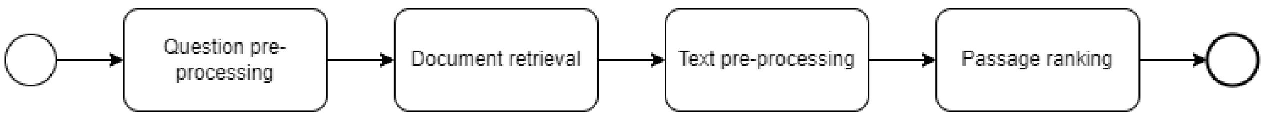


Figure 4.10: Generic process of the recommender system

4.1 Question pre-processing

In a medical research papers recommender system, input as natural language question can be pre-processed in two different ways including question classification and keywords identification. For question classification, questions can be grouped into different topics in medical fields such as gene, diagnosis, epidemiology, etc. [16] or according to their question type, for instance Yes/ No, Factoid or Summary question [59]. While categorizing question into various topics can assist finding the sources of information that focus on each topic, categorizing it into distinct question types can be used subsequently to specify ways to form answers to the questions. Meanwhile, keywords identification might be useful for documents retrieval step. Apart from extracting main subjects of the question using different named-entity recognition techniques, in medical field those subjects can also be matched with medical entities using biomedical terminology systems such as UMLS, SemRep or MetaMap.

For our recommender system, question classification does not play a significant role as the recommender neither have to choose data sources based on topics nor define question types itself. The only database that we consider is from MEDLINE from PubMed and the recommender will suggest relevant passages that related to the question without trying to form an answer. Meanwhile, keyword identification is a crucial step as PubMed search engine retrieve the most number of results with input as keywords instead of natural language question. At Bloom Diagnostics, the step of keywords identification is still performed manually by medical researchers. After attempting to map the input question with different biomedical terminology systems include UMLS, SemRep and MetaMap, the author of the thesis has come to the conclusion that those terminology systems still can not beat medical researchers in enriching list of relevant keywords, thus the lists of identified keywords is shorter and fewer potentially relevant articles are found. Therefore, rather than automatically extracting keywords and sending those keywords to retrieve articles, the current application at Bloom Diagnostics only suggests a list of keywords based on UMLS, SemRep, and MetaMap and allows medical researchers to modify and provide their own input. In the dataset used for evaluation, the keyword identification step is done in advance by medical researchers from their previous researches to maximize the number of returned articles.

4.2 Documents retrieval

Depending on the medical research paper databases, retrieving documents from those databases can be done differently. Among four recommender systems for medical research papers that were investigated in Chapter 2, SemBioNLQA and askMEDLINE use database of MEDLINE from PubMed for document retrieval thus utilizing its Entrez Programming Utilities to search through PubMed. Meanwhile, AskHERMES collect data from various sources and use BM25 model to retrieve relevant documents. BioMine use Solr search platform for their document retrieval task which allow input from different databases and use Solr's search methods to search through texts.

At Bloom Diagnostics, PubMed is the main search engine that is used for document retrievals, thus using Entrez Programming Utilities is their optimal choice as Entrez Programming Utilities does not require database storage space and can be integrate into their web application easily. The current search application at Bloom Diagnostics also uses the technique of sending list of keywords to Entrez to extract potentially relevant documents.

4.3 Text pre-processing

In order to be pre-processed, texts need to be scraped first. Get inspired by the company's current strategy of scraping articles for the titles, abstracts, and full-texts, Entrez Programming Utilities is used for the extraction of the titles and abstracts. Obtaining the full text, on the other hand, is more challenging and requires more effort. To obtain full-text, typically web scraping frameworks such as Selenium or Beautiful Soup can be employed. PubMed search engine gathers information from several medical databases, and for each returned result, offers various links that direct visitors to the source database's full-text content. The fact that full-text articles come from different sources cause difficulties in scraping them automatically. It might require a lot of efforts and is time-consuming to figure out the format of each of the sources' website PubMed us for scraping. Therefore, for the purpose of evaluating the methodologies of this thesis, the author decides to scrap full-text articles manually in order to obtain the maximum number of full-text articles for testing.

After being scraped, each document will be saved as a separate .txt file, ready for pre-processing and extracting. Drawing inspiration from the pre-processing methodologies employed by AskHERMES, the thesis adopts a similar approach to handle tables, lists and passages. For each table, the text content within each row or column is combined with the respective header and caption text to form a distinct passage. Each list is treated as a tree structure, where every non-leaf branch node (including the root node) generates a separate passage. This is achieved by collapsing all the nodes within the branch and incorporating the textual information with the caption text. Furthermore, section names are combined with any paragraphs found within the corresponding section. This is based on the recognition that section titles in articles carry significant semantic value that may not be explicitly expressed in the narrative paragraphs. [14]. Titles and abstracts will begin with the words "Title" and "Abstract" to distinguish them from other sections and paragraphs. This tic-tac is used since we would like to know whether an article can be included or excluded based on its title or abstract. "Keywords", "Topic", and "Issue section" parts are typically omitted during pre-processing, despite the fact that they give the most vital information about the articles. In our work, "Keywords", "Topic", and "Issue section" sections are merged with the title during pre-processing to improve the semantic meaning of the title. Elements such as "Contact information" and "Claims of conflict of interest" that are confusing or unnecessary are eliminated. Unnecessary spaces, tabs, and question marks have also been omitted. As the .tsv file used to feed BERT uses the tab character as a separator, deleting tabs from all passages prevents the file from being formatted incorrectly. Additionally, the model throws errors when those paragraphs contain a question mark, therefore they are eliminated during the pre-processing step.

After full-texts are pre-processed and splitted into passages, a unique ID was provided for each passage. As the passages will eventually be tracked back to their articles, the passage ID must include the article's ID in PubMed. Each passage within an article requires a unique identifier to store the section's location for later review. Thus, the theis uses ID format for passages as [article ID]_[passage location]. Passages' locations are numbered in ascending order beginning with 1. Thanks to this, each passage has its own unique number and can be traced back with relative articles. Additionally, questions are assigned with unique ID as input for the model.

4.4 Passage ranking

In order to find the passages in the document that help answer the question the most, passages are ranked in term of relevancy to the question. Passage here can be a title, an abstract, a single paragraph, a table or a list within the documents. Numerous NLP models and techniques can be used for this task, ranging from tradition models like Okapi BM25 that employ bag-of-words technique to neural network models that learn the representation of passages and question to rank passages based on learned features. Among those neural network models, BERT is a model that attracts a great deal of attention for surpassing other models when it comes to dealing with various NLP issues. In addition, the source code for BERT with different configurations and some of its fine-tuned models are publicly accessible for further study. Another point to consider is that the key aspect of a recommender system for research papers is assessing the relevance between the search query and the returned articles and ranking them from most to least relevant. Thus we choose to focus on this part of passage ranking for the scope of this thesis.

4.4.1 Comparing different models

The thesis will concentrate mostly on document/ passage ranking/ re-ranking and analyze various ways. Documents and passages are ranked or re-ranked based on their relevance to the question, allowing the users to discover the passages or documents that may provide the best answer to the question using various models. The employed models range from the common document ranking algorithm BM25 to the more advanced BERT and BioBERT models. Nogueira et al. [50] introduced a fine-tuned model of BERT for re-ranking passages, which can be used here. The paper "Passage Re-ranking with BERT" [50] describes their achievements on the Passage Re-Ranking challenge of MS MARCO [6]. The challenge presented competitors with the top 1.000 passages as retrieved by BM25 and required them to re-rank these passages by relevance [49]. While the idea of using BM25 model for initial ranking then using a more computational-costly model for re-ranking is from the MS MARCO challenge, using BERT as re-ranking model is proposed by Nogueira et al. [50] in their paper. Based on those ideas, the thesis proposes various combinations for ranking/re-ranking passages, from BM25 alone to BM25 combined with BERT and BioBERT in various settings. After that, the thesis will test those settings on different datasets using different evaluation criteria to determine the model that have the balance between running time and efficiency. Furthermore, for each dataset, there are two variants with varying numbers of articles to study how each model responds to datasets of varying sizes. The settings of models consist of:

- Using BM25 to extract top 1,000 relevant passages, then use BERT-Large to re-rank and extract top 50 most relevant passages

- Using BM25 to extract top 1,000 relevant passages, then use BERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 1,000 relevant passages, then use BioBERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 100 relevant passages, then use BERT-Large to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 100 relevant passages, then use BERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 100 relevant passages, then use BioBERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 50 most relevant passages

Note that while the source code to reproduce BERT fine-tuning for passage ranking/ re-ranking is available on GitHub, to the best of our knowledge, fine-tuning BioBERT for passage ranking/ re-ranking has not been performed previously. Therefore, the thesis will present the main steps for fine-tuning BioBERT in the next section.

4.4.2 Fine-tuning BioBERT

The BioBERT fine-tuning process consists of the construction of a medical dataset containing medical questions with multiple passages as candidate answers, the configuration of all fine-tuning parameters, and the report on execution's status. In order to compare BERT and BioBERT's performance on the task, BioBERT must be fine-tuned separately.

Preparing the dataset By giving input in the medical domain with the same format and weights of BioBERT instead of BERT and specifying the relevant parameters, it is possible to reuse the code for fine-tuning BERT for document re-ranking with BioBERT.

To maximize effectiveness in fine-tuning, it is crucial to have a training dataset that closely resembles the dataset used in production. The process of fine-tuning will modify the final layer of the BioBERT model, enabling it to perform downstream tasks. The datasets used to fine-tune BERT came from the MS MARCO competition [6]. MS MARCO has a collection of datasets for deep learning with various tasks, such as question answering, passage ranking/re-ranking, key phrase extraction, and conversational search, etc. MS MARCO provided the dataset used by Nogueira et al. [50] for the passage re-ranking task. The data structure for fine-tuning for passage re-ranking is a list of questions, each of which is accompanied by a list of 1,000 candidate passages ranked by BM25. One passage out of a total of 1,000 is recognized as relevant, while the remaining 999 are marked as irrelevant.

A new medical dataset for passage re-ranking based on credible resources is constructed. Among the common medical datasets, BioASQ provides a trustworthy dataset for Question Answering. Initiated in 2018, the BioASQ challenge [1] aims to promote the state-of-the-art in large-scale biomedical semantic indexing and Question Answering by offering datasets for a variety of tasks that researchers can utilize to discover optimal solutions. The accuracy of these biomedicine-specific datasets was evaluated by domain specialists. Each dataset includes a question and its corresponding answer. By retrieving all BioASQ datasets and removing duplicates, the thesis generated a list of 4,320 question-answer pairs. Both questions and answers have been indexed, and the answer will be marked as relevant to the question.

Theoretically, in order to accomplish the passage rating objective, the BM25 model should have identified the 999 most relevant responses for each question. The thesis also employs the BM25 algorithm to accommodate the data format of the analogous problem for fine-tuning BERT. All 999 of these passages were marked as irrelevant to the question. From the original dataset, the thesis created a new dataset with 4,320,000 inputs for re-ranking passages. This approximately 20GB-sized dataset was processed using Colab Pro Premium GPU for approximately six hours. In general, the larger the training dataset, the more precise the resulting model. However, as the limited capacity of training equipment, the thesis only uses BM25 to extract the best 199 passages instead of 999, resulting in 200 candidate passages per question. Only one passage was labeled relevant, while the remaining 199 were as irrelevant. Thus, the size of the dataset is reduced by one-fifth, to around 864,000 lines and roughly 4GB.

This dataset is separated with the ratio 14:2:1 into training set, development set, and test set. The 3,500-question training set was utilized for the main training. During the process of creating checkpoints for fine-tuning, a 500-question development set was utilized to evaluate model improvement. Each of these sets includes a distinct .qrels file to show the relevance of these responses. The 230-question test set was utilized to evaluate the performance of the model. Even though the MS MARCO utilized a different rate for splitting, which is around 40:30:30, the thesis decided to employ a different ratio [6] for more data for training set. The specifications of each set are as follows:

- Training set consists of 3,500 questions with 697,523 lines
- Development set consists of 500 questions with 99,641 lines
- Evaluation set consists of 230 questions with 45,935 lines

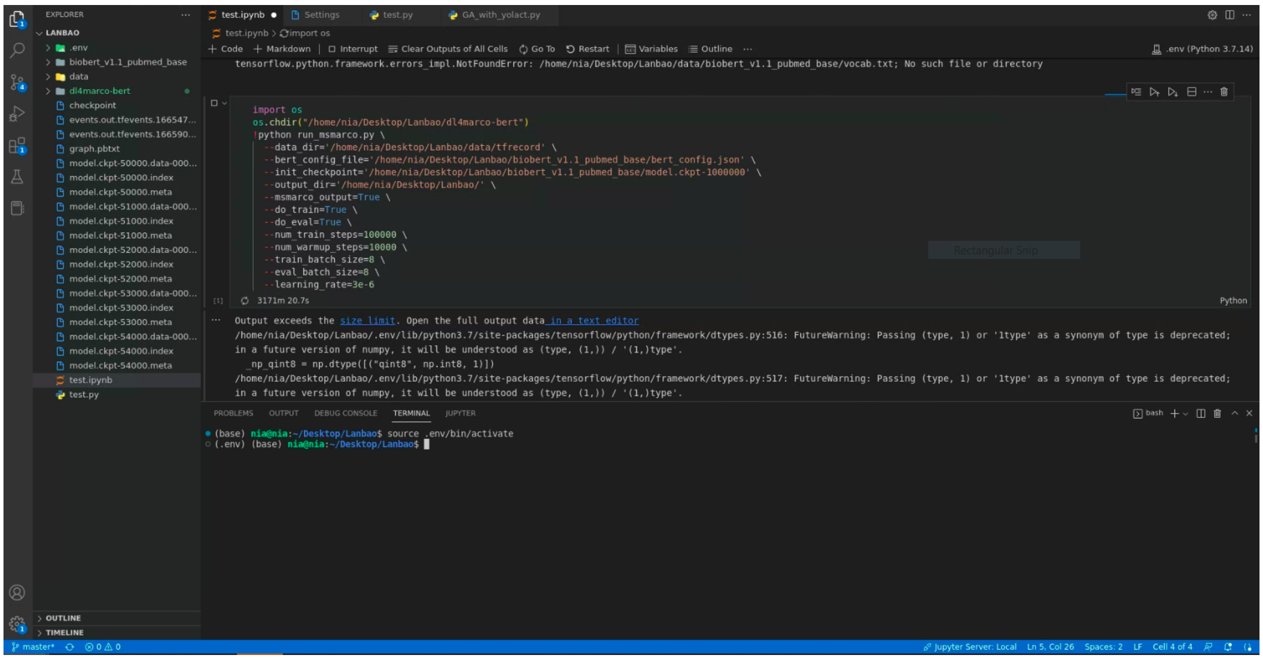
After establishing distinct sets, the dataset was converted to TFRecord so that BioBERT could subsequently use it for fine-tuning. Maximum query length and maximum sequence length are set to 64 and 512 characters, respectively. 200 evaluation passages were used to match the training dataset.

Fine-tuning setup and running The initial weight for the model to be fine-tuned is BioBERT Base v1.1 plus one million articles from PubMed. This is the most recent version of BioBERT, which uses the same language as BERT-Base-Cased. The model was recommended by the BioBERT authors. After obtaining the initial weight and data set, the process of fine-tuning was initiated. BioBERT Base was fine-tuned on a computer with hardware of core i9 10900, 128GB RAM, and three NVIDIA RTX 3090 24GB GPUs. From the 14th to the 18th of October 2022, five days were devoted to fine-tuning. The parameter for fine-tuning is as follows:

- number of training steps: 100,000
- number of warmup steps: 10,000
- training batch size: 8

- evaluation batch size: 8
- learning rate: 3e-6

The number of training steps, which is 100,000, represents the number of checkpoints that will be created during training. The first 10,000 steps of these 100,000 points will be taught at the minimum learning rate. Starting at step 10,001, the model will employ the learning rate of 3e-6. Note that the number of training steps, the number of warm-up steps, and the learning rate were all set to the same value for MS MARCO's fine-tuned BERT for equivalent results. However, utilizing a training batch size and evaluation batch size of 128 consistently results in a memory overload error, thus the batch size must be lowered to a maximum of eight for the training to function. The model saved a new checkpoint point every 1,000 steps and destroyed all checkpoints prior to the most recent three. After five days, the model had generated only 54,000 checkpoints or nearly half of the total. Due to restricted server access, training had to be halted and the weights at 54,000 checkpoints was taken for running evaluation. The fine-tuned BioBERT for passage re-ranking was considered complete.



```
LANBAO
  .env
  biobert_v1.1_pubmed_base
  data
  dl4marco-bert
    checkpoint
    events.out.tfevents.166547...
    events.out.tfevents.166590...
    graph.pbtxt
    model.ckpt-50000.data-000...
    model.ckpt-50000.index
    model.ckpt-50000.meta
    model.ckpt-51000.data-000...
    model.ckpt-51000.index
    model.ckpt-51000.meta
    model.ckpt-52000.data-000...
    model.ckpt-52000.index
    model.ckpt-52000.meta
    model.ckpt-53000.data-000...
    model.ckpt-53000.index
    model.ckpt-53000.meta
    model.ckpt-54000.data-000...
    model.ckpt-54000.index
    model.ckpt-54000.meta
    test.ipynb
    test.py
```

```
tensorflow.python.framework.errors_impl.NotFoundError: /home/nia/Desktop/Lanbao/data/biobert_v1.1_pubmed_base/vocab.txt; No such file or directory
```

```
import os
os.chdir("/home/nia/Desktop/Lanbao/dl4marco-bert")
!python run_msmarco.py \
  --data_dir='/home/nia/Desktop/Lanbao/data/tfrecord' \
  --bert_config_file='/home/nia/Desktop/Lanbao/biobert_v1.1_pubmed_base/bert_config.json' \
  --init_checkpoint='/home/nia/Desktop/Lanbao/biobert_v1.1_pubmed_base/model.ckpt-1000000' \
  --output_dir='/home/nia/Desktop/Lanbao/' \
  --msmarco_output=True \
  --do_train=True \
  --do_eval=True \
  --num_train_steps=100000 \
  --num_warmup_steps=10000 \
  --train_batch_size=8 \
  --eval_batch_size=8 \
  --learning_rate=3e-6
```

[1] 3171m 20.7s

```
Output exceeds the size limit. Open the full output data in a text editor
/home/nia/Desktop/Lanbao/.env/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
/home/nia/Desktop/Lanbao/.env/lib/python3.7/site-packages/tensorflow/python/framework/dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synonym of type is deprecated;
in a future version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

```
(base) nia@nia:~/Desktop/Lanbao$ source .env/bin/activate
(.env) (base) nia@nia:~/Desktop/Lanbao$
```

Figure 4.11: Checkpoints created during fine-tuning BioBERT

5 Evaluation

This chapter will cover the experimental design and outcomes in depth. While the first section describes the test sets used to evaluate the experiments, the next section describes the experimental setup design and results of all evaluations.

In order to validate the models combinations mentioned above, the thesis will utilize three previously collected datasets from the Bloom Diagnostics Ferritin project. Each dataset is accompanied with a question, a search query, and a list of extracted articles. The list of papers will also include the medical research team's assessment of the article's relevance for evaluation purpose. By utilizing actual datasets for evaluation, this thesis aims to provide a realistic perspective on how these methodologies work in real-world use cases. When it comes to constructing a system for recommending research papers, the validation will also be viewed in accordance with the situation of the company and other small to medium businesses.

The evaluation is performed using the typical measures for evaluating classification models including accuracy, error, sensitivity, specificity, precision, F1 score, and running time. F1 score is the harmonic of precision and was utilized to evaluate different BERT variants. Running time plays a significant role in the design of a recommender system. Normal users will expect the findings to arrive within a few seconds. In comparison, for a more complex problem that requires specialized knowledge, for instance in the medical domain, researchers might accept no more than a few minutes of processing time. In order to be considered applicable, these techniques must also satisfy the condition of running time. From those evaluations, the thesis will determine the optimal model/ models combination that may be applied to the company by evaluating several methodologies on various datasets. The subsequent chapter provides details for the evaluation.

5.1 Datasets

In order to evaluate different model combinations on datasets of different sizes, the thesis first extracts essential data from three distinct datasets, including questions, full-texts of returned papers, and researchers' evaluations of each paper's relevance. The thesis then considers those three datasets as "small" datasets and enriches them to create "large" versions that can be compared to the "small" ones.

5.1.1 Small datasets

The thesis utilizes a dataset comprised of three smaller datasets from the Ferritin project at Bloom Diagnostics to capture the implementation and effectiveness of different models in real-world circumstances. The datasets are part of a project that Bloom Diagnostics' medical researchers have done to validate their Ferritin assay test. Each dataset includes a question, a related search query, and a collection of retrieved PubMed articles. Relevance evaluations from the medical team are also included in the test sets. Each dataset's evaluation was performed manually by researchers and thoroughly investigated at various levels of supervision. A multi-level supervision and strict process in evaluation helps ensure the reliability for evaluations of the datasets. Medical datasets are scarce as they must originate from reputable sources and be evaluated by domain experts. In this instance, the dataset satisfies this condition since it was performed by a team of professionals with the necessary degrees and certificates.

Note that due to the company's unique characteristics, it only serves customers in certain locations and age groups. For example, studies that focus on the Asia region will be eliminated because the company has not yet expanded to this region. Additionally, research on children under the age of 18 will be eliminated, as these tests are only advised for adults. This is a company-specific need that does not apply to more general situations. While at Bloom Diagnostics the articles focus on those populations will be labeled as irrelevant, in the scope of this thesis we decided to omitted them instead in order to maintain the evaluation's simplicity and primary goals of ranking relevance without concerning about user groups. In addition, because the approaches would like to focus on the ranking of full-text articles, any articles without accessible full-text will be excluded. The PMC database and a few other journals, such as Elservier, SpringerLink, Oxford Academic, etc., are some of the sources for full-text articles in PubMed. Note that in all datasets, the Humans and English filters are used to receive only publications written in English and pertaining to the Human species. Table 5.4 provides an overview of all test sets.

	Question	Number of retrieved articles	Number of relevant articles
Dataset 1	Which infections increase Ferritin levels? What is the consequence of that?	29	14
Dataset 2	What is the benefit of measuring ferritin in vegetarians? What are the daily iron intake recommendations for vegetarians?	17	9
Dataset 3	What happens to iron levels during exercise in subjects with healthy serum ferritin levels?	22	3

Table 5.4: All test sets

Dataset 1 The dataset seeks to answer the question: "Which infections increase Ferritin levels? What is the consequence of that? " The question associates with the topic infection in Ferritin project topic. There are 34 articles returned. Five articles were eliminated, two due to the target group being underage and three for lack of full-text availability. 29 full-text articles were scraped for evaluation. Among those, 14 out of 29 are labeled relevant.

Dataset 2 The dataset seeks to answer the question: "What is the benefit of measuring ferritin in vegetarians? What are the daily iron intake recommendations for vegetarians? " The question associates with the topic vegetarian in Ferritin project. There are 40 articles returned. 23 articles were eliminated, ten due to the target group being underage, two for irrelevant populations and eleven due to the lack of full-text availability. 17 full-text articles were scraped for evaluation. Among those, nine out of 17 are labeled relevant.

Dataset 3 The dataset seeks to answer the question: "What happens to iron levels during exercise in subjects with healthy serum ferritin levels? " The question associates with the topic exercise in Ferritin project. There are 52 articles returned. Five articles were eliminated, two due to the target group being underage, eleven for not relevant populations and 17 for the lack of available full-text. 22 full-text articles were scraped for evaluation. Among those, three out of 22 are labeled relevant.

5.1.2 Large datasets

In addition to smaller datasets, which were the original datasets from Bloom Diagnostics, the intention of this thesis is to also evaluate the performance of these models on larger datasets. The smaller and larger datasets must be comparable. To generate the larger dataset, we add a number of additional full-text articles to the smaller one, bringing the total number of full-text articles in each larger dataset to 90. Thus, while the smaller datasets contain around 850 to 1,100 passages, the larger datasets are three to four times larger. These full texts are also a part of the Ferritin project, which aims to challenge the system's ability to discover relevant articles. However, as the added articles are potential responses to a different question, they are considered irrelevant to the current question. Additionally, since all additional articles are marked as irrelevant, the number of relevant articles remains unchanged. While the larger Dataset 1 contains 90 articles with 14 relevant articles, the larger Dataset 2 have 90 articles with nine relevant articles and the larger Dataset 3 have 90 articles with only three relevant articles.

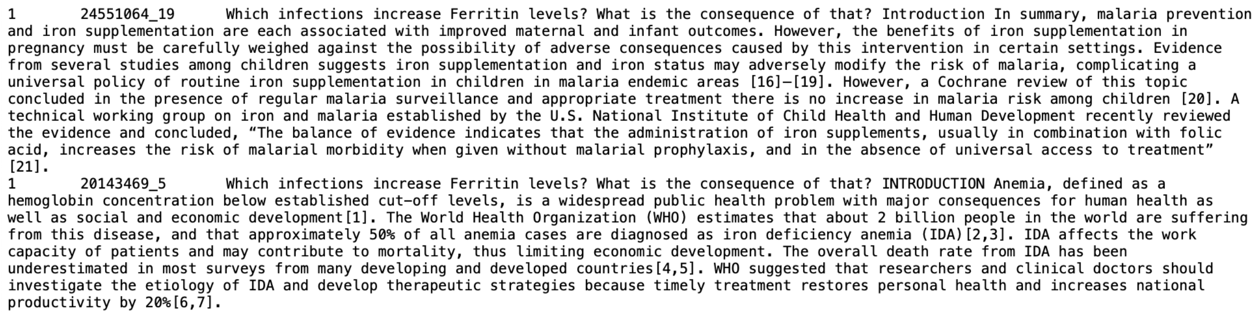
5.2 Setup

For each question, there are two datasets, the smaller one being the company's original dataset and the larger one being an expanded version of the smaller one. There are six datasets to evaluate in all. There are seven possible models settings for each dataset:

- Using BM25 to extract top 1,000 relevant passages, then use BERT-Large to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 1,000 relevant passages, then use BERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 1,000 relevant passages, then use BioBERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 100 relevant passages, then use BERT-Large to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 100 relevant passages, then use BERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 100 relevant passages, then use BioBERT-Base to re-rank and extract top 50 most relevant passages
- Using BM25 to extract top 50 most relevant passages

This produces 42 unique outcomes for evaluations and comparisons. The purpose of the thesis is to generate as many outcomes as possible in order to gain a broader understanding of the methodologies employed. The evaluations were conducted using a Colab Pro Premium GPU, which is similar to an Nvidia V100 or A100 GPU [3]. This GPU is as powerful as a server that small and medium-sized businesses can utilize to run their internal software. Consequently, the duration of each evaluation will be comparable to actual firm outcomes.

The evaluation procedure is as follows. First, the full text of potential articles are retrieved from PubMed. These full texts were then pre-processed as stated in Chapter 4 and divided into passages, with each passage consisting of a title, abstract, or paragraph. In order to compute relevancy between the candidate passages and the natural language question, both information is then supplied to BM25, BERT, and BioBERT using the following format: [question ID] [Passage ID] [Question content] [Passage content]. Figure 5.12 displays an example of the input format. First, all data is sent through the BM25 algorithm in order to retrieve the 50, 100, or 1,000 most relevant paragraphs. Depending on the settings, the output from BM25 is re-ranked using BERT-Base, BERT-Large, or BioBERT-Base to obtain the top 50 passages. These leading 50 paragraphs were extracted and evaluated.



```
1       24551064_19      Which infections increase Ferritin levels? What is the consequence of that? Introduction In summary, malaria prevention
and iron supplementation are each associated with improved maternal and infant outcomes. However, the benefits of iron supplementation in
pregnancy must be carefully weighed against the possibility of adverse consequences caused by this intervention in certain settings. Evidence
from several studies among children suggests iron supplementation and iron status may adversely modify the risk of malaria, complicating a
universal policy of routine iron supplementation in children in malaria endemic areas [16]–[19]. However, a Cochrane review of this topic
concluded in the presence of regular malaria surveillance and appropriate treatment there is no increase in malaria risk among children [20]. A
technical working group on iron and malaria established by the U.S. National Institute of Child Health and Human Development recently reviewed
the evidence and concluded, “The balance of evidence indicates that the administration of iron supplements, usually in combination with folic
acid, increases the risk of malarial morbidity when given without malarial prophylaxis, and in the absence of universal access to treatment”
[21].
1       20143469_5       Which infections increase Ferritin levels? What is the consequence of that? INTRODUCTION Anemia, defined as a
hemoglobin concentration below established cut-off levels, is a widespread public health problem with major consequences for human health as
well as social and economic development[1]. The World Health Organization (WHO) estimates that about 2 billion people in the world are suffering
from this disease, and that approximately 50% of all anemia cases are diagnosed as iron deficiency anemia (IDA)[2,3]. IDA affects the work
capacity of patients and may contribute to mortality, thus limiting economic development. The overall death rate from IDA has been
underestimated in most surveys from many developing and developed countries[4,5]. WHO suggested that researchers and clinical doctors should
investigate the etiology of IDA and develop therapeutic strategies because timely treatment restores personal health and increases national
productivity by 20%[6,7].
```

Figure 5.12: Data input examples

As the passage IDs contain information regarding passage location and article ID, these passages then are linked to their respective articles. If an article has at least one passage among the top 50 most relevant passages, it is considered relevant to the question. If not, the article is considered irrelevant. Thus, an article's value will be binary: relevant or irrelevant. If there are multiple passages in an article that stay in the top 50, the passage with the highest rank determines the article's final rank. Typically, there are two methods for calculating the rank of an article based on passages ranking: picking the highest rank or summing the ranking scores of all the passages as the article's score. Here, the thesis employs the first strategy, as the information needed to answer the question typically comes from a single paragraph, which should have the highest rank. The rank is utilized for showing search results, which display the most relevant articles and its content at the top in descending order of relevance.

Relevant articles are considered as positive, while irrelevant articles are considered as negative. The thesis calculates True Positive, False Positive, True Negative, and False Negative using the confusion matrix. These statistics are used to determine accuracy, error, sensitivity, specificity, precision, and F1 score. Running time is the total time for both ranking and re-ranking, which includes the time required to run BM25 and BERT or BioBERT in the first six cases and BM25 in the seventh case. The results are described in detail in the following section.

5.3 Results

In this section, the main result of each evaluation is introduced and compared with other results based on a variety of indexes such as accuracy, error, sensitivity, specificity, precision, F1 score, and running time.

5.3.1 Dataset 1

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.72	0.66	0.69	0.66	0.59	0.66	0.52
Error	0.28	0.34	0.31	0.34	0.41	0.34	0.48
Sensitivity	0.50	0.43	0.43	0.50	0.50	0.57	0.50
Specificity	0.93	0.87	0.93	0.80	0.67	0.73	0.53
Precision	0.88	0.75	0.86	0.70	0.58	0.67	0.50

Table 5.5: Classification evaluation metrics of Dataset 1 (small)

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	125.73	44.36	93.18	21.77	10.12	10.56	0.04
F1 score (%)	63.7	54.6	57.3	58.3	53.7	61.6	50.0

Table 5.6: Running time and F1 score of Dataset 1 (small)

Small dataset Tables 5.5 and 5.6 summarize the performance of all models on Dataset 1 (small). Using BM25 to extract top 1,000 passages and re-ranking by BERT-Large for the top 50 passages has the highest accuracy and lowest error rate (0.72 and 0.28) for Dataset 1 (small). Using only BM25 to extract top 50 passages yields the lowest accuracy and highest error, which are 0.52 and 0.48 respectively. Using BM25 to extract top 1,000 followed by BERT-Base for the top 50, using BM25 to extract top 100 followed by BERT-Large for the top 50, and using BM25 to extract top 100 followed by BioBERT-Base for the top 50 have the same accuracy of 0.66. Using BM25 to extract top 100 then re-ranking by BioBERT-Base has the highest sensitivity, meaning it has the highest ratio of true positives to total positives in the data. Using BM25 to extract top 1,000 then BERT-Large for the top 50 and BM25 to extract top 1,000 then BioBERT-Base for the top 50 have the highest specificity of 0.93. In terms of precision, using BM25 to extract top 1,000 then re-ranking by BERT-Large for top 50 perform the best, followed by using BM25 for extracting top 1,000 then re-ranking by BioBERT-Base for top 50. Precision for using only BM25 to extract top 50 passages is the lowest value with only 0.5. Using BM25 to extract top 1,000 then re-ranking using BERT-Large for the top 50 takes approximately two minutes, whereas using only BM25 to extract top 50 just takes 0.04 seconds. The highest F1 scores goes to the BM25 for extracting top 1,000 then using BERT-Large.

	BM25: 1,000 $\rightarrow$ BERT-Large: 50	BM25: 1,000 $\rightarrow$ BERT-Base: 50	BM25: 1,000 $\rightarrow$ BioBERT-Base: 50	BM25: 100 $\rightarrow$ BERT-Large: 50	BM25: 100 $\rightarrow$ BERT-Base: 50	BM25: 100 $\rightarrow$ BioBERT-Base: 50	BM25: 50
Accuracy	0.86	0.86	0.83	0.81	0.79	0.81	0.78
Error	0.14	0.14	0.17	0.19	0.21	0.19	0.22
Sensitivity	0.36	0.36	0.21	0.29	0.29	0.29	0.21
Specificity	0.95	0.95	0.95	0.91	0.88	0.91	0.88
Precision	0.56	0.56	0.43	0.36	0.31	0.36	0.25

Table 5.7: Classification evaluation metrics of Dataset 1(large)

	BM25: 1,000 $\rightarrow$ BERT-Large: 50	BM25: 1,000 $\rightarrow$ BERT-Base: 50	BM25: 1,000 $\rightarrow$ BioBERT-Base: 50	BM25: 100 $\rightarrow$ BERT-Large: 50	BM25: 100 $\rightarrow$ BERT-Base: 50	BM25: 100 $\rightarrow$ BioBERT-Base: 50	BM25: 50
Running time (s)	122.23	56.14	91.01	21.04	10.01	12.8	0.09
F1 score (%)	43.8	43.8	28.2	32.1	30.0	32.1	22.8

Table 5.8: Running time and F1 score of Dataset 1 (large)

Large dataset Tables 5.7 and 5.8 summarize the performance of all models on Dataset 1 (large). Using BM25 to extract top 1,000 then BERT-Large or BERT-Base to re-rank yields the same results and has the highest results among all models in terms of accuracy, sensitivity, specificity and precision. BERT-Large requires two minutes for re-ranking 1,000 passages, while BERT-Base requires less than one minute to perform the same task. Using BM25 only has the lowest values for accuracy, sensitivity, specificity, and precision, and the greatest value for error, compared to other models. In terms of accuracy and specificity, however, the difference of using BM25 only is 0.08 and 0.07 points respectively compared to best-performed models. Using BM25 to extract top 1,000, followed by BioBERT-Base, has the highest specificity as of using BM25 to extract top 1,000, followed by BERT-Large or BERT-Base for the top 50. Using BM25 to extract top 1,000 then applying BERT-Large, BERT-Base outperform other models in terms of F1 score, which have F1 score value of 43.8%.

Comparing Dataset 1 small and large Comparing Dataset 1 version small and large, it is evident that all models perform better with a smaller dataset in term on F1 score. The difference between the best and worst outcomes of F1 score for smaller and larger datasets is 13.7 and 11.0 percentage points, respectively. While BioBERT-Base outperforms BERT-Base in all settings for the small dataset, this is not the case for the large dataset as the performance between BioBERT-Base and BERT-Base fluctuates. The running time of different models on smalle and large dataset is equivalent. The rationale for this is that BM25 require almost the same time to rank 1,000 or 5,000 passages. After the initial ranking by BM25, all models were left with the same amount of passages. Thus, the running times are comparable. In terms of accuracy, all models perform better on larger datasets than on smaller ones, although for precision different models perform better on the small dataset compared to the large dataset.

5.3.2 Dataset 2

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.76	0.88	0.71	0.82	0.82	0.76	0.76
Error	0.24	0.12	0.29	0.18	0.18	0.24	0.24
Sensitivity	0.89	0.89	0.56	0.78	0.78	0.67	0.67
Specificity	0.62	0.88	0.88	0.88	0.88	0.88	0.88
Precision	0.73	0.89	0.83	0.88	0.88	0.86	0.86

Table 5.9: Classification evaluation metrics of Dataset 2 (small)

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	127.55	44.80	87.45	21.81	10.07	8.63	0.02
F1 score (%)	80.2	89.0	66.9	82.7	82.7	75.0	75.3

Table 5.10: Running time and F1 score of Dataset 2 (small)

Small dataset Tables 5.9 and 5.10 summarize the performance of all models on the second small dataset. For Dataset 2 (small), using BM25 to extract top 1,000 and BERT-Base for the top 50 provides the maximum accuracy. Among all models, using BM25 to extract top 1,000 then using BioBERT-Base has the lowest accuracy at 0.71. Using BM25 to extract top 1,000 then using BERT-Large or BERT-Base to re-rank has the highest sensitivity, while using BM25 to extract top 1,000 then BioBERT-Base for re-ranking has the lowest sensitivity of 0.56. Other than using BM25 to extract top 1,000 and BERT-Large for top 50, all other models have a specificity of 0.88. Using BM25 to extract top 1,000 and then BERT-Base to re-rank performs the best in terms of precision, whereas using BM25 to extract top 1,000 and then BERT-Large is the poorest in the field. Consequently, BM25 to extract top 1,000 followed by

BERT-Base for top 50 has the greatest F1 score, followed by BM25 to extract top 100 and BERT-Large or BERT-Base to re-rank.

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.91	0.94	0.94	0.91	0.91	0.94	0.88
Error	0.09	0.06	0.06	0.09	0.09	0.06	0.12
Sensitivity	0.56	0.78	0.67	0.44	0.44	0.56	0.44
Specificity	0.95	0.96	0.98	0.96	0.96	0.99	0.93
Precision	0.56	0.70	0.75	0.57	0.57	0.83	0.40

Table 5.11: Classification evaluation metrics of Dataset 2 (large)

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	126.68	43.81	84.91	21.41	9.96	9.98	0.09
F1 score (%)	56.0	73.8	70.8	49.7	50.0	66.9	41.9

Table 5.12: Running time and F1 score of Dataset 2 (large)

Large dataset Tables 5.11 and 5.12 summarize the performance of all models on the second large dataset. All models exhibit quite high accuracy on the large dataset, ranging from 0.88 to 0.94. After extracting top 100 passages using BM25, BioBERT-Base performs better than BERT-Base and even BERT-Large with regards to accuracy and specificity. Using BM25 to extract top 1,000 and BERT-Base for the top 50 achieves the maximum result for sensitivity. Accuracy and precision are lowest when using only BM25, at 0.88 and 0.40 respectively. All models do exceptionally well in terms of specificity. Specificity is 0.99 when using BM25 to extract top 100, and 0.98 when using BM25 to extract top 1,000 and BioBERT-Base for the top 50. In general, using BM25 for extracting top 1,000 followed by BERT-Base for top 50 and BM25 for extracting top 1,000 followed by BioBERT-Base perform the best for F1 score, whilst other models perform not as good.

Comparing Dataset 2 small and large In addition to having a higher accuracy, all models also return higher specificity in the large dataset compared to the small one. Using BM25 to extract top 1,000 and BERT-Base for the top 50 offers the greatest accuracy in both versions of Dataset 2. Using BM25 for extracting top 1,000 then BERT-Base for top 50 has a better value in terms of sensitivity, specificity, and precision than using BM25 for extracting top 1,000 then BERT-Large to re-rank in both small and large of Dataset 2. Meanwhile using BM25 for top 100 then BERT-Base for top 50 and using BM25 for top 100 then BERT-Large to re-rank have the same value in both datasets. Using BM25 to extract top 1,000 and BERT-Base for the top 50 yields the best F1 score in both versions of Dataset 2.

5.3.3 Dataset 3

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.55	0.59	0.59	0.50	0.59	0.50	0.55
Error	0.45	0.41	0.41	0.50	0.41	0.50	0.45
Sensitivity	0.67	0.67	1.00	0.67	0.67	0.67	0.67
Specificity	0.53	0.58	0.53	0.47	0.58	0.47	0.53
Precision	0.18	0.20	0.25	0.17	0.20	0.17	0.18

Table 5.13: Classification evaluation metrics of Dataset 3 (small)

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	127.57	44.03	46.02	21.96	10.64	8.24	0.02
F1 score (%)	28.4	30.8	40.0	27.1	30.8	26.7	28.3

Table 5.14: Running time and F1 score of Dataset 3 (small)

Small dataset Tables 5.13 and 5.14 summarize the performance of all models on Dataset 3 (small). Using BM25 to extract top 1,000 then BERT-Large to re-rank and using simply BM25 yields the same performance score in evaluation metric while BM25 only requires much shorter time to process compared to using BM25 to extract top 1,000 then BERT-Large to re-rank. Using BM25 to extract top 1,000, followed by BERT-Base or BioBERT-Base for the top 50, and BM25 to extract top 100, followed by BERT-Base for the top 50, has the greatest accuracy of 0.59. Meanwhile, using BM25 to extract top 100, then BERT-Large to re-rank, and using BM25 to extract top 100, then BioBERT-Base to re-rank produce identical results. Using BM25 to extract top 1,000, followed by BioBERT-Base accurately predicted three true positive articles and zero false negative articles therefore its sensitivity reached the maximum value of 1.00. Precision is generally low, ranging between 0.17 and 0.25. Using BM25 to extract top 1,000 then BioBERT-Base yields the highest F1 score of 40.0, 9.2 percentage points higher than the second-highest score obtained by using BM25 to extract top 1,000 or 100 then using BERT-Base for the top 50.

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.89	0.88	0.88	0.87	0.89	0.87	0.55
Error	0.11	0.12	0.12	0.13	0.11	0.13	0.45
Sensitivity	0.67	0.33	0.33	0.33	0.33	0.33	0.33
Specificity	0.90	0.90	0.90	0.89	0.91	0.89	0.58
Precision	0.18	0.10	0.10	0.09	0.11	0.09	0.11

Table 5.15: Classification evaluation metrics of Dataset 3 (large)

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	128.34	44.33	51.45	21.26	10.07	9.01	0.08
F1 score (%)	28.4	15.3	15.4	14.1	16.5	14.3	16.5

Table 5.16: Running time and F1 score of Dataset 3 (large)

Large dataset Tables 5.15 and 5.16 summarize the performance of all models on the Dataset 3 (large). The variances in accuracy among most models range from 0.87 to 0.89, which is not a wide range except for using BM25 only. Using BM25 for extracting top 1,000 then BERT-Large to re-rank has a sensitivity of 0.67, while the other models have a sensitivity of 0.33 since they only forecast the relevance of one out of three articles. In addition, it is not a significant difference in specificity across most combinations except for using BM25 only. Using BM25 for extracting top 1,000 then BERT-Large to re-rank yields the best F1 score, while using BM25 for top 100 then BERT-Large to re-rank yields the lowest F1 score of 14.1.

Comparing Dataset 3 small and large In both versions of Dataset 3, accuracy and specificity do not fluctuate excessively except for using only BM25. Using BM25 to extract top 1,000 and BERT-Large to re-rank have the same F1 score in both datasets; however, while 28.4 is the highest F1 score in the large dataset, using BM25 to extract top 1,000 and BioBERT-Base has the highest value in term of F1 score in the small dataset. Using BM25 to extract top 1,000, then BioBERT-Base achieves the highest F1 score on the small dataset, while its score on the large dataset is in the middle of the range.

		BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	Q1	0.72	0.66	0.69	0.66	0.59	0.66	0.52
	Q2	0.76	0.88	0.71	0.82	0.82	0.76	0.76
	Q3	0.55	0.59	0.59	0.5	0.59	0.5	0.55
Error	Q1	0.28	0.34	0.31	0.34	0.41	0.34	0.48
	Q2	0.24	0.12	0.29	0.18	0.18	0.24	0.24
	Q3	0.45	0.41	0.41	0.50	0.41	0.50	0.45
Sensitivity	Q1	0.50	0.43	0.43	0.50	0.50	0.57	0.50
	Q2	0.89	0.89	0.56	0.78	0.78	0.67	0.67
	Q3	0.67	0.67	1.00	0.67	0.67	0.67	0.67
Specificity	Q1	0.93	0.87	0.93	0.80	0.67	0.73	0.53
	Q2	0.62	0.88	0.88	0.88	0.88	0.88	0.88
	Q3	0.53	0.58	0.53	0.47	0.58	0.47	0.53
Precision	Q1	0.88	0.75	0.86	0.70	0.58	0.67	0.50
	Q2	0.73	0.89	0.83	0.88	0.88	0.86	0.86
	Q3	0.18	0.20	0.25	0.17	0.20	0.17	0.18

Table 5.17: Classification evaluation metrics of each small dataset

5.3.4 Comparing results for small datasets

Tables 5.17 and 5.18 demonstrate the performances of all models on each small dataset. In comparison among all small datasets, Dataset 2 always have the highest accuracy and lowest error among all datasets. In terms of sensitivity, all models perform better on Dataset 1 than the Dataset 2. Using BM25 to extract top 1,000 followed by BioBERT-Base has the lowest sensitivity values for both in Dataset 1 and 2, which are 0.43 and 0.56 respectively. The models' performance on Dataset 3 is much unstable. Additionally, the precision of Dataset 1 and 2 is significantly higher than that of Dataset 3. Regarding F1 score, the performance on Dataset 2 is superior to the performance on Dataset 1 and 3 across all settings. It is important to note that the running time of BioBERT-Base for re-ranking varies across different questions, whereas other approaches perform quite consistently, with only a fraction of a second difference.

		BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	Q1	125.73	44.36	93.18	21.77	10.12	10.56	0.04
	Q2	127.55	44.80	87.45	21.81	10.07	8.63	0.02
	Q3	127.57	44.03	46.02	21.96	10.64	8.24	0.02
F1 score (%)	Q1	63.7	54.6	57.3	58.3	53.7	61.6	50.0
	Q2	80.2	89.0	66.9	82.7	82.7	75.0	75.3
	Q3	28.4	30.8	40.0	27.1	30.8	26.7	28.3

Table 5.18: Running time and F1 score of each small dataset

5.3.5 Comparing results for large datasets

Tables 5.19 and 5.20 demonstrate the performances of all models on each large dataset. Overall, the accuracy of all models for all large datasets is quite high, ranging from 0.78 to 0.94, which the exception of using only BM25 on Dataset 3 which has accuracy of 0.55. Consequently, errors are low. The second dataset has a higher sensitivity than the other two datasets, except for employing BM25 to extract top 1,000 and BERT-Large to re-rank in Dataset 3. All models in the large datasets had specificity values more than 0.88, with the exception of using BM25 alone in the Dataset 3. Precision, in contrast, is low, particularly for Dataset 3. Dataset 2 has the highest precision score of 0.83 by using BM25 to extract top 100 followed by BioBERT-Base. Regarding running time, the BioBERT-Base has the broadest range compared to other models.

5.3.6 Comparing results for small and large datasets

Tables 5.21 and 5.23 take into account the performances on all small versions, while tables 5.22 and 5.24 show the performances on all large versions. Taking the average of all small and large datasets, it is evident that the smaller dataset has higher error, sensitivity, and precision, while the larger dataset has higher accuracy and specificity. The accuracy of all models for small datasets ranges from 0.61 for BM25 alone to 0.71 when using BM25 to extract top 1,000 and BERT-Base for the top 50, while the accuracy for large datasets ranges from 0.74 to 0.89. The sensitivity of the models on the small datasets are not far from each other, with a 0.08-point difference between the lowest and highest value of sensitivity. The distance of sensitivity value in the larger dataset is wider, 0.33 from using only BM25 to 0.53 of using BM25 for extracting top 1,000 then BERT-Large to re-rank. Using BM25 to extract top 1,000 and BERT-Base or BioBERT-Base for the top 50 provides the maximum specificity in both versions small and large, whereas BM25 alone yields the lowest point of 0.65 and 0.80. Using BM25 to extract top 1,000 then BioBERT-Base has the greatest value for precision across all large datasets, while for small datasets it has the second highest value.

		BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	Q1	0.86	0.86	0.83	0.81	0.79	0.81	0.78
	Q2	0.91	0.94	0.94	0.91	0.91	0.94	0.88
	Q3	0.89	0.88	0.88	0.87	0.89	0.87	0.55
Error	Q1	0.14	0.14	0.17	0.19	0.21	0.19	0.22
	Q2	0.09	0.06	0.06	0.09	0.09	0.06	0.12
	Q3	0.11	0.12	0.12	0.13	0.11	0.13	0.45
Sensitivity	Q1	0.36	0.36	0.21	0.29	0.29	0.29	0.21
	Q2	0.56	0.78	0.67	0.44	0.44	0.56	0.44
	Q3	0.67	0.33	0.33	0.33	0.33	0.33	0.33
Specificity	Q1	0.95	0.95	0.95	0.91	0.88	0.91	0.88
	Q2	0.95	0.96	0.98	0.96	0.96	0.99	0.93
	Q3	0.90	0.90	0.90	0.89	0.91	0.89	0.58
Precision	Q1	0.56	0.56	0.43	0.36	0.31	0.36	0.25
	Q2	0.56	0.70	0.75	0.57	0.57	0.83	0.4
	Q3	0.18	0.10	0.10	0.09	0.11	0.09	0.11

Table 5.19: Classification evaluation metrics of each Large Dataset

5.3.7 Comparing different models

Tables 5.25 and 5.26 give an overview of how all settings perform on all datasets. Using BM25 to extract top 1,000 and BERT-Base for the top 50 performs the best in terms of accuracy across all datasets. The performance of other BERT and BioBERT models are relatively close to each other in term of accuracy, while BM25 alone has an accuracy of 0.67, which is 0.13 points worse than of using BM25 for extracting top 1,000 followed by BERT-Base. In terms of sensitivity, using BM25 alone has the lowest value among all models while using BM25 to extract top 1,000, BERT-Large has the highest value of 0.61. Using BM25 to extract top 1,000 then BERT-Base or BioBERT-Base for the top 50 has the highest specificity value, followed by using BM25 to extract top 100 then BERT-Large to re-rank with a value of 0.82. Using BM25 to extract top 1,000, then using BioBERT-Base has the greatest precision value. Using BM25 to extract top 1,000 then BERT-Large to re-rank is the most effective model considering F1 score. All BERT and BioBERT-based approaches outperform BM25 by 4.9 to 12.1 percentage points in term of F1 score. BioBERT-Base performs inconsistently compared to BERT-Based, doing better when re-ranking 100 passages in term of F1 score but worse when re-ranking top 1,000 passages from BM25. The running time of BioBERT-Base techniques is inconsistent as well. Last but not least, BERT-Large and BERT-Base behave fairly similarly for all measures.

		BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	Q1	122.23	56.14	91.01	21.04	10.01	12.80	0.09
	Q2	126.68	43.81	84.91	21.41	9.96	9.98	0.09
	Q3	128.34	44.33	51.45	21.26	10.07	9.01	0.08
F1 score (%)	Q1	43.8	43.8	28.2	32.1	30.0	32.1	22.8
	Q2	56.0	73.8	70.8	49.7	50.0	66.9	41.9
	Q3	28.4	15.3	15.4	14.1	16.5	14.3	16.5

Table 5.20: Running time and F1 score of each large dataset

5.3.8 Medical experts' evaluation

To re-evaluate the data, test sets for the first two questions, "Which infections increase Ferritin levels? What is the consequence of that?" and "What is the benefit of measuring ferritin in vegetarians? What are the daily iron intake recommendations for vegetarians?" was evaluated by medical experts. The medical researcher's team at Bloom Diagnostics was asked to assess the passages returned by the models to determine whether or not these passages offer information that aids in answering the question. The thesis is only able to double-check the outcomes of questions that were processed by BM25 to extract top 1,000 passages and BERT-Large for the top 50 passages and the small datasets due to the time-consuming and expensive nature of the assessment process.

19 out of 50 returned passages are marked my researchers as relevant for all passages selected by BM25 and BERT-Large for the first question. Relevant indicates that the paragraph contains information that can be used to answer the question. Regarding the second question, 27 out of 50 passages are labeled relevant. It is noteworthy to note that for both outcomes, all relevant paragraphs come from previously selected relevant articles. It means there are no relevant passages from irrelevant articles among such passages. Despite the fact that just two datasets confirm the relationship between the relevance of articles and their passages, the practice of utilizing article relevance to determine passage relevance still has a foundation.

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.68	0.71	0.66	0.66	0.67	0.64	0.61
Error	0.32	0.29	0.34	0.34	0.33	0.36	0.39
Sensitivity	0.69	0.66	0.66	0.65	0.65	0.64	0.61
Specificity	0.69	0.78	0.78	0.72	0.71	0.69	0.65
Precision	0.60	0.61	0.65	0.58	0.55	0.57	0.51

Table 5.21: Classification evaluation metrics on average of all small datasets

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.89	0.89	0.88	0.86	0.86	0.87	0.74
Error	0.11	0.11	0.12	0.14	0.14	0.13	0.26
Sensitivity	0.53	0.49	0.40	0.35	0.35	0.39	0.33
Specificity	0.93	0.94	0.94	0.92	0.92	0.93	0.80
Precision	0.43	0.45	0.43	0.34	0.33	0.43	0.25

Table 5.22: Classification evaluation metrics on average of all large datasets

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	126.95	44.40	75.55	21.85	10.28	9.14	0.03
F1 score (%)	57.4	58.1	54.7	56.0	55.7	54.4	51.2

Table 5.23: Running time and F1 score on average of all small datasets

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	125.75	48.09	75.79	21.24	10.01	10.60	0.09
F1 score (%)	42.7	44.3	38.1	32.0	32.2	37.8	27.1

Table 5.24: Running time and F1 score on average of all large datasets

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Accuracy	0.78	0.80	0.77	0.76	0.77	0.76	0.67
Error	0.22	0.20	0.23	0.24	0.24	0.24	0.33
Sensitivity	0.61	0.58	0.53	0.50	0.50	0.52	0.47
Specificity	0.81	0.86	0.86	0.82	0.81	0.81	0.72
Precision	0.52	0.53	0.54	0.46	0.44	0.50	0.38

Table 5.25: Classification evaluation metrics of all settings

	BM25: 1,000 → BERT-Large: 50	BM25: 1,000 → BERT-Base: 50	BM25: 1,000 → BioBERT-Base: 50	BM25: 100 → BERT-Large: 50	BM25: 100 → BERT-Base: 50	BM25: 100 → BioBERT-Base: 50	BM25: 50
Running time (s)	126.35	46.25	75.67	21.54	10.15	9.87	0.06
F1 score (%)	50.1	51.2	46.2	44.0	44.0	46.1	39.1

Table 5.26: Running time and F1 score of all settings

6 Conclusion

The final chapter includes a discussion of the major findings of the evaluations, followed by a discussion of the limitations of the current approach and some suggestions for future research.

6.1 Discussion

First, it is noticeable that the settings of using BM25 to extract the top 1,000 passages and BERT Base for the top 50 passages perform the best in terms of F1 score and accuracy. Regarding precision, the performance of BM25 to extract the top 1,000 passages, then using BERT, and BioBERT is comparable, while BioBERT has the best result among all other settings. Nonetheless, the difference is only between 0.01 and 0.06 points.

BM25 cannot compete in terms of accuracy, sensitivity, specificity, precision, and F1 score (%) with the BERT re-ranking models. However, among the datasets of this thesis, the differences in the measures are only because of a single article. Yet, due to the intuitive nature of the BM25 method, the running time is substantially less than that of other methods and is close to being instantaneous.

In the majority of settings, utilizing BM25 for extracting top 1,000 passages performs better than its variants that use BM25 to extract top 100 passages and then re-rank. In terms of accuracy, using BM25 for extracting the top 1,000 passages is 0.01 to 0.03 points more accurate than using BM25 for extracting the top 100 passages. Regarding sensitivity, the disparity is greater. Using BM25 to extract the top 1,000 and BERT-Large or BERT-Base to re-rank improves the results far more than using BM25 to extract the top 100 passages. Meanwhile, using BM25 to extract the top 1,000 passages, BioBERT-Base is only 0.01 points better in terms of sensitivity than its 100 passages equivalent when compared using BM25. It demonstrates that the initial ranking task performed by BM25 has an effect on the final outcomes. The more leniently the BM25 algorithm filters a list of potential passages, the more precise the models.

In general, the performance of BioBERT-Base is comparable to the performance of fine-tuned BERT-Base. However, the performance of BioBERT-Base varies among all datasets. In comparison to BERT-Large and BERT-Base, BioBERT-Base has inconsistent performance. Despite BioBERT's evaluation showing that it outperforms BERT in domain-specific tasks [41], using BioBERT does not enhance the results in our evaluation. The below-average performance of fine-tuned BioBERT could be attributable to a number of factors, including the incompleted fine-tuning procedure.

Nonetheless, all BERT and BioBERT re-ranking models outperform BM25 alone in all used datasets. Using BM25 to extract the top 1,000 passages and BERT-Base for the top 50 passages improves F1 score by twelve points compared to using BM25 alone. However, the experiment also demonstrates that the difference only comes from a single article in each category.

Based on the gathered findings and observations, it can be inferred that combining BM25 and BERT variants can offer advantages in both running time and accuracy. For rapid results, BM25 can be utilized to provide the initial results as it provides users with results with less waiting time. While users investigate returned results, BERT variants can be employed in the background to re-rank and present additional potentially relevant articles. The result of utilizing the BM25 algorithm alone will be displayed first. The algorithm then identifies more potentially relevant articles or passages to increase accuracy.

Overall, the utility of BERT models depends on the capacity of the company and the nature of the task. When resources are available, BERT and its derivatives have the potential to produce a superior outcome. This is true especially in the case of Bloom Diagnostics, where the quantity of retrieved articles is restricted and any additional relevant piece of information or article is greatly appreciated. However, compared to BM25, BERT training is quite costly. Even if the process of fine-tuning was promised to be inexpensive, it still requires a significant amount of time and work.

In general, running time of BERT variants is longer than the running time of BM25. Although BERT is more resource-intensive and time-consuming to train and process, in this particular case, it only predicted one additional relevant article compared to BM25 in our experiments. Therefore, despite the fact that BERT can promise a better outcome, corporations can continue to employ traditional algorithms like BM25 and can still achieve an acceptable result in a resource-efficient way.

6.2 Limitations

The approach and evaluation process presented in this thesis has some limitations, which will be discussed in the following.

First, the medical dataset that we created for fine-tuning BioBERT might affect BioBERT's performance. Due to the limited server resources required to run the entire dataset, fine-tuning BioBERT can only be done from approximately 4GB of data out of the full 20GB dataset that we created for fine-tuning. The model can only learn to rank the top 200 passages as opposed to the 1,000 required by the approaches. Given the fact that fine-tuning BERT for passage re-ranking requires a training dataset of approximately 13 GB [6], fine-tuned BioBERT model for passage re-ranking that was created by us may not be as good as expected.

The limited access to a powerful server hindered the completion of the process of fine-tuning. As just 54 percent of the fine-tuning has been completed, it may not have reached its optimal points. In addition, while studying the process of fine-tuning, the models do not make optimal use of all available resources. Non-updated transformer libraries, in this case Tensorflow, may result in an inefficient operating process. However, because the server is inaccessible after a certain number of days, the author of the thesis cannot reach an additional improvement model. The fact that the fine-tuning BERT Cloud TPU fit a training batch size of 128 [50] whilst the used server can only load a batch size of eight is likely one of the reasons why fine-tuning BioBERT by us takes longer than fine-tuning BERT for passage re-ranking. In addition, when performing fine-tuning, it is advisable to experiment with various parameters and test continually until the ideal parameters are identified. Based on the settings of linked models and the server's capability, the thesis comes up with the initial feasible parameters, which may not be the optimal parameter values.

Automatically generating larger datasets based on the available, manually created datasets from Bloom Diagnostics for evaluation has certain drawbacks. While it helps evaluate model performance and assess their effectiveness on a larger scale, the larger dataset lacks verification from domain experts. While assuming that the added articles for larger datasets are irrelevant, it is possible that they have been mislabeled. As a result, the reported performance of all models on large datasets represents the lower end of their performance ranges.

As mentioned in the Literature Review chapter, the development of ranking algorithms for recommender systems has been a longstanding issue that has been addressed in numerous studies. Even the BM25 or BERT models have a lot of different variants that can be used. The BM25 algorithm exists in different variants, which aim to improve the results by incorporating other calculations such as the novel scoring functions or the longest common subsequence score in AskHERMES. Since BERT has demonstrated its outstanding performance and potential in the NLP field, different models based on transformers have been proposed. Recently, ChatGPT [20] has caught a lot of attention in the NLP field for its capacity to handle different kinds of questions and generate answers based on contexts. ChatGPT is an NLP system developed by Open AI that is based on a deep learning model called GPT (in different versions). As a successor of BERT, XLNet algorithm [77] claims to outperform BERT. XLNet offers the potential to optimize the anticipated likelihood across all conceivable permutations of the factorization sequence. It surpasses the constraints of BERT due to its autoregressive structure. Another effort in creating an Open-source Multilingual Language Model named BLOOM [60] also caught attention in the scientific world. BLOOM is based solely on the decoder Transformer architecture and comprises numerous sources across 46 natural languages and 13 programming languages [60].

6.3 Future works

Taking into account the limits and peculiarities of a recommender system for medical research papers, the following can still be accomplished.

BioBERT could be fine-tuned on more potent servers. Using the entire dataset for re-ranking the top 1,000 passages might potentially increase the learning capability of BioBERT by providing a sufficient number of learning examples. After the fine-tuning process has been completed, the parameters for fine-tuning BioBERT can also be re-considered to determine if there have been any improvements. BioBERT can be fine-tuned numerous times with different settings to compare the loss functions of each fine-tuning, if necessary. Updating the current code to Tensorflow v2 is one of the potential solutions to the problem of server resources being underutilized. Upgrading Tensorflow version has been applied in attempts to improve the prediction time of several BERT versions and has been successful in decreasing the execution time from hours to minutes and seconds.

In addition to BERT, other new models and methodologies can be used for the task of constructing a recommender system. As a result of the gaps between research activities and real-world efficacy, NLP techniques have shown numerous improvement opportunities. There are ongoing efforts to develop innovative new techniques. XLNet, SMITH, and BLOOM are three of the most promising models in tackling the problem of passage/document ranking/re-ranking.

Essential to the development of a comprehensive recommender system for research papers is the construction of pipelines to scrape full text from diverse sources. NCBI does not provide an API for scraping full-text articles; consequently, other methods must be employed to achieve this objective. The full texts on PubMed come from several sources and journals, each with its own format, which must be explored separately. From the PMC database, which is owned by NCBI and is the primary source for full-text from PubMed, additional sources can be investigated. The greater the scraping attempts from various sources, the more amount of available full-text articles can be found. Also, the recommender system still requires methods for automatically scrapping full-text and pre-process data to be completed. Additionally, a User Interface is needed to show results to users.

In addition to employing several models for ranking and re-ranking, the application can also use user comments to improve search results. There are numerous features to consider that fall into several categories including Social, Content, or Thread features. Social features are determined by the level of interaction between users and the system, such as the proportion of returned results that are read by users. Content features seek to find articles or search terms that are highly connected with users' actions on the system, such as the occurrence of a search word in articles or passages. Thread features record the user's activity with the thread thus far, such as whether the user initiated the thread [7].

References

[1] BioASQ: The challenge. http://bioasq.org/, Accessed: 09/06/2023.

[2] BM25 Reference, 2022. https://docs.vespa.ai/en/reference/bm25.html, Last accessed on 2022-12-23.

[3] Colab's 'pay as you go' offers more access to powerful NVIDIA compute for machine learning, 2022. https://blog.tensorflow.org/2022/09/colabs-pay-as-you-go-offers-more-access-to-powerful-nvidia-compute-for-machine-learning.html#:~:text=Paid%20Colab%20users%20can%20now%20choose%20between%20a,typically%20NVIDIA%20V100%20or%20A100%20Tensor%20Core%20GPUs., Last accessed on 2022-12-27.

[4] dl4marco-bert, 2022. https://github.com/nyu-dl/dl4marco-bert, Last accessed on 2022-12-27.

[5] Okapi BM25, 2022. https://en.wikipedia.org/wiki/Okapi_BM25, Last accessed on 2022-12-23.

[6] MS MARCO, 2023. https://microsoft.github.io/msmarco/, Last accessed on 2023-1-1.

[7] Douglas Aberdeen, Ondrey Pacovsky, and Andrew Slater. The learning behind gmail priority inbox. 2010.

[8] Daniel E Acuna, Zijun Yi, Kartik Nagre, and Priya Matnani. EILEEN: A recommendation system for scientific publications and grants. *arXiv preprint arXiv:2110.09663*, 2021.

[9] Hayda Almeida, Ludovic Jean-Louis, and Marie-Jean Meurs. Mining biomedical literature: an open source and modular approach. In *Canadian Conference on Artificial Intelligence*, pages 168–179. Springer, 2016.

[10] Sanjay Arora, Anne L Peters, Elizabeth Burner, Chun Nok Lam, and Michael Menchine. Trial to examine text message–based mhealth in emergency department patients with diabetes (TExT-MED): A randomized controlled trial. *Annals of emergency medicine*, 63(6):745–754, 2014.

[11] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[12] Riccardo Bellazzi and Blaz Zupan. Predictive data mining in clinical medicine: current issues and guidelines. *International journal of medical informatics*, 77(2):81–97, 2008.

[13] Charles Bridges-Webb. A computer summary for general practice medical records: MEDSUM. *J. Fam. Pract*, 23:389–392, 1986.

[14] YongGang Cao, Feifan Liu, Pippa Simpson, Lamont Antieau, Andrew Bennett, James J Cimino, John Ely, and Hong Yu. AskHERMES: An online question answering system for complex clinical questions. *Journal of biomedical informatics*, 44(2):277–288, 2011.

[15] Hsinchun Chen, Sherrilynne S Fuller, Carol Friedman, and William Hersh. *Medical informatics: knowledge management and data mining in biomedicine*, volume 8. Springer Science & Business Media, 2006.

[16] Travers Ching, Daniel S Himmelstein, Brett K Beaulieu-Jones, Alexandr A Kalinin, Brian T Do, Gregory P Way, Enrico Ferrero, Paul-Michael Agapow, Michael Zietz, Michael M Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387, 2018.

[17] Krzysztof J Cios and G William Moore. Uniqueness of medical data mining. *Artificial intelligence in medicine*, 26(1-2):1–24, 2002.

[18] Penny Coppernoll-Blach. Quertle: the conceptual relationships alternative search engine for PubMed. *Journal of the Medical Library Association*, 99(2):176–177, 2011.

[19] Sarah Cruchet, Arnaud Gaudinat, and Célia Boyer. Supervised approach to recognize question type in a QA system for health. *Studies in Health Technology and Informatics*, 136:407, 2008.

[20] Jianyang Deng and Yijia Lin. The benefits and challenges of ChatGPT: An overview. *Frontiers in Computing and Intelligent Systems*, 2(2):81–83, 2022.

[21] Priya Desai, Natalie Telis, Ben Lehmann, Keith Bettinger, Jonathan K Pritchard, and Somalee Datta. SciReader: a cloud-based recommender system for biomedical literature. *BioRxiv*, page 333922, 2018.

[22] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[23] Bloom Diagnostics. About Bloom, 2022. `https://www.linkedin.com/company/bloomdiagnostics/`, Last accessed on 2022-10-14.

[24] Bloom Diagnostics. Technology, 2022. `https://bloomdiagnostics.com/en_AT/technology`, Last accessed on 2022-10-14.

[25] Anna Divoli and Teresa K Attwood. BioIE: extracting informative sentences from the biomedical literature. *Bioinformatics*, 21(9):2138–2139, 2005.

[26] Andreas Doms and Michael Schroeder. GoPubMed: exploring PubMed with the gene ontology. *Nucleic acids research*, 33(suppl_2):W783–W786, 2005.

[27] Richard Easty and Nikolay Nikolov. Client-side integration of life science literature resources. *Bioinformatics*, 25(23):3194–3196, 2009.

[28] Alfred D Eaton. HubMed: a web-based biomedical literature search interface. *Nucleic acids research*, 34(suppl_2):W745–W747, 2006.

[29] Robin Featherstone and Denise Hersey. The quest for full text: an in-depth examination of Pubget for medical searchers. *Medical Reference Services Quarterly*, 29(4):307–319, 2010.

[30] Jean-Fred Fontaine, Adriano Barbosa-Silva, Martin Schaefer, Matthew R Huska, Enrique M Muro, and Miguel A Andrade-Navarro. MedlineRanker: flexible ranking of biomedical literature. *Nucleic acids research*, 37(suppl_2):W141–W146, 2009.

[31] Paul Fontelo, Fang Liu, and Michael Ackerman. ask MEDLINE: a free-text, natural language query tool for MEDLINE/PubMed. *BMC medical informatics and decision making*, 5(1):1–6, 2005.

[32] Paul Fontelo, Fang Liu, Sergio Leon, Abrahamane Anne, and Michael Ackerman. PICO linguist and BabelMeSH: Development and partial evaluation of evidence-based multilanguage search tools for Medline/Pubmed. *Studies in Health Technology and Informatics*, 129(1):817, 2007.

[33] Martin Gerner, Farzaneh Sarafraz, Casey M Bergman, and Goran Nenadic. BioContext: an integrated text mining system for large-scale extraction and contextualization of biomolecular events. *Bioinformatics*, 28(16):2154–2161, 2012.

[34] S Larry Goldenberg, Guy Nir, and Septimiu E Salcudean. A new era: artificial intelligence and machine learning in prostate cancer. *Nature Reviews Urology*, 16(7):391–403, 2019.

[35] Marti A Hearst, Anna Divoli, Harendra Guturu, Alex Ksikes, Preslav Nakov, Michael A Wooldridge, and Jerry Ye. BioText Search Engine: beyond abstract search. *Bioinformatics*, 23(16):2196–2197, 2007.

[36] William Hersh. *Information retrieval: a health and biomedical perspective*. Springer Science & Business Media, 2008.

[37] Karsten Hokamp and Kenneth H Wolfe. PubCrawler: keeping up comfortably with PubMed and GenBank. *Nucleic acids research*, 32(suppl_2):W16–W19, 2004.

[38] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W Cohen, and Xinghua Lu. PubMedQA: A dataset for biomedical research question answering. *arXiv preprint arXiv:1909.06146*, 2019.

[39] Jung-jae Kim, Piotr Pezik, and Dietrich Rebholz-Schuhmann. MedEvi: retrieving textual evidence of relations between biomedical concepts from MEDLINE. *Bioinformatics*, 24(11):1410–1412, 2008.

[40] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[41] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[42] James Lewis, Stephan Ossowski, Justin Hicks, Mounir Errami, and Harold R Garner. Text similarity: an alternative way to search MEDLINE. *Bioinformatics*, 22(18):2298–2304, 2006.

[43] Daniel Loureiro and Alípio Mário Jorge. Medlinker: Medical entity linking with neural representations and dictionary matching. In *European Conference on Information Retrieval*, pages 230–237. Springer, 2020.

[44] Zhiyong Lu. PubMed and beyond: a survey of web tools for searching biomedical literature. *Database*, 2011, 2011.

[45] Michael Muin, Paul Fontelo, Fang Liu, and Michael Ackerman. SLIM: an alternative Web interface for MEDLINE/PubMed searches–a preliminary study. *BMC medical informatics and decision making*, 5(1):1–9, 2005.

[46] National Library of Medicine. Introduction to the unified medical language system, 2022. `https://www.nlm.nih.gov/bsd/disted/video/clin_info/umls.html#:~:text=The%20Unified%20Medical%20Language%20System%20(UMLS)%20is%20a%20set%20of,anatomy%2C%20genes%2C%20and%20more.`, Last accessed on 2022-10-15.

[47] National Library of Medicine. Pubmed's homepage, 2022. `https://pubmed.ncbi.nlm.nih.gov/`, Last accessed on 2022-10-15.

[48] National Library of Medicine. Semrep, 2022. `https://lhncbc.nlm.nih.gov/ii/tools/SemRep_SemMedDB_SKR/SemRep.html`, Last accessed on 2022-10-15.

[49] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *choice*, 2640:660, 2016.

[50] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019.

[51] Tiago Nunes, David Campos, Sérgio Matos, and José Luís Oliveira. BeCAS: biomedical concept recognition services and visualization. *Bioinformatics*, 29(15):1915–1916, 2013.

[52] National Library of Medicine. Metamap - a tool for recognizing umls concepts in text, 2022. `https://lhncbc.nlm.nih.gov/ii/tools/MetaMap.html`, Last accessed on 2022-10-15.

[53] Carolina Perez-Iratxeta, Peer Bork, and Miguel A Andrade. XplorMed: a tool for exploring MEDLINE abstracts. *Trends in biochemical sciences*, 26(9):573–575, 2001.

[54] Maksim V Plikus, Zina Zhang, and Cheng-Ming Chuong. PubFocus: semantic MEDLINE/PubMed citations analytics through integration of controlled biomedical dictionaries and ranking algorithm. *BMC bioinformatics*, 7(1):1–15, 2006.

[55] Sudharsan Ravichandiran. *Getting Started with Google BERT: Build and train state-of-the-art natural language processing models using BERT*. Packt Publishing Ltd, 2021.

[56] Dietrich Rebholz-Schuhmann, Harald Kirsch, Miguel Arregui, Sylvain Gaudan, Mark Riethoven, and Peter Stoehr. EBIMed—text crunching to gather facts for proteins from MEDLINE. *Bioinformatics*, 23(2):e237–e244, 2007.

[57] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

[58] Indra Neil Sarkar, Ryan Schenk, Holly Miller, and Catherine N Norton. LigerCat: using "MeSH Clouds" from journal, article, or gene citations to facilitate the identification of relevant biomedical literature. In *AMIA Annual Symposium Proceedings*, volume 2009, page 563. American Medical Informatics Association, 2009.

[59] Mourad Sarrouti and Said Ouatik El Alaoui. SemBioNLQA: A semantic biomedical question answering system for retrieving exact and ideal answers to natural language questions. *Artificial intelligence in medicine*, 102:101767, 2020.

[60] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[61] Lisa Seyfried, David A Hanauer, Donald Nease, Rashad Albeiruti, Janet Kavanagh, and Helen C Kales. Enhanced identification of eligibility for depression research using an electronic medical record search engine. *International journal of medical informatics*, 78(12):e13–e18, 2009.

[62] Dikshant Shahi. Apache Solr: an introduction. In *Apache Solr*, pages 1–9. Springer, 2015.

[63] Mir S Siadaty, Jianfen Shu, and William A Knaus. Relemed: sentence-level search engine with relevance score for the MEDLINE database of biomedical articles. *BMC medical informatics and decision making*, 7(1):1–11, 2007.

[64] Neil R Smalheiser, Dean P Fragnito, and Eric E Tirk. Anne O'Tate: Value-added PubMed search engine for analysis and text mining. *PloS one*, 16(3):e0248335, 2021.

[65] David J States, Alex S Ade, Zachary C Wright, Aaron V Bookvich, and Brian D Athey. MiSearch adaptive PubMed search tool. *Bioinformatics*, 25(7):974–976, 2009.

[66] Philippe Thomas, Johannes Starlinger, Alexander Vowinkel, Sebastian Arzt, and Ulf Leser. GeneView: a comprehensive semantic search engine for PubMed. *Nucleic acids research*, 40(W1):W585–W591, 2012.

[67] Andrew Trotman and David Keeler. Ad hoc IR: not much room for improvement. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1095–1096, 2011.

[68] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, pages 58–65, 2014.

[69] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*, 2019.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[71] Jiannan Wang, Inci Cetindil, Shengyue Ji, Chen Li, Xiaohui Xie, Guoliang Li, and Jianhua Feng. Interactive and fuzzy search: a dynamic way to explore MEDLINE. *Bioinformatics*, 26(18):2321–2327, 2010.

[72] Jacob White. PubMed 2.0. *Medical Reference Services Quarterly*, 39(4):382–387, 2020.

[73] Claes Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pages 1–10, 2014.

[74] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[75] Yasunori Yamamoto and Toshihisa Takagi. Biomedical knowledge navigation by literature clustering. *Journal of biomedical informatics*, 40(2):114–130, 2007.

[76] Yasunori Yamamoto, Atsuko Yamaguchi, Hidemasa Bono, and Toshihisa Takagi. Allie: a database and a search service of abbreviations and long forms. *Database*, 2011, 2011.

[77] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[78] Takashi Yoneya and Hiroshi Mamitsuka. PURE: a PubMed article recommendation system based on content-based filtering. *Genome informatics*, 18:267–276, 2007.

[79] Hwanjo Yu, Taehoon Kim, Jinoh Oh, Ilhwan Ko, Sungchul Kim, and Wook-Shin Han. Enabling multi-level relevance feedback on PubMed by integrating rank learning into DBMS. In *Proceedings of the third international workshop on Data and text mining in bioinformatics*, pages 43–50, 2009.

Acronyms

AI Artificial Intelligence. 1

BERT Bidirectional Encoder Representation from Transformer. iii, 3, 7, 13, 15–17, 20, 21

CRF Conditional Random Field. 9

GO Gene Ontology. 4, 5, 7

IDF Inverse Document Frequency. 12, 13

MeSH Medical Subject Heading. 4–7

ML Machine Learning. 9, 10

MLM Masked language modeling. 16, 17

NCBI National Center for Biotechnology Information. 3

NER Named Entity Recognition. 19

NLM United States National Library of Medicine. 3, 8, 23

NLP Natural Language Processing. 1, 2, 9, 13, 19, 20, 27, 49

NSP Next Sentence Prediction. 16, 18

SVM Support Vector Machine. 6, 7, 9

TF-IDF Term Frequency-Inverse Document Frequency. 4, 5, 8, 10

UMLS Unified Medical Language System. 4–8, 10–12