Enhanced PubMed Search Strategies: Integrating Neural Models and MeSH Terms for Precision in Literature Retrieval

Leandra Budau

***Abstract*—This paper introduces a novel model aimed at enhancing Systematic Literature Review (SLR) Boolean search queries by integrating relevant Medical Subject Heading (MeSH) terms. Using three pretrained transformer-based models, the approach involves ChatGPT for query generation, BERTMeSH for MeSH term suggestions, and a BERT-based sentence transformer for semantic comparison. The model is implemented in Google Colab and evaluated on CLEF TAR 2017 and 2018 datasets, employing metrics such as precision, recall, and f-measure. Results reveal competitive precision and f-measure, but a notable deficiency in recall, indicating a need for further exploration. A comparative analysis with the Smooth Operator Model underscores the distinct nature of the two approaches, with the proposed model outperforming in precision and f-measure but trailing in recall. Suggestions for improvement include exploring manual query translations, refining the semantic comparison model through domain-specific training, and investigating the optimal number of MeSH term additions.**

***Index Terms*—Information retrieval, Natural language processing, Medical Subject Headings (MeSH), Systematic Literature Review (SLR), Boolean search queries.**

I. Introduction

SYSTEMATIC Literature Reviews are important academic studies that are often considered the most valid form of evidence for medical decision making [1]. They involve deciding on a research question, searching for research articles relevant to the question, determining which of the articles meet the study criteria, and then using them to answer the research question. An important step in an SLR is called the study search, which involves gathering all possibly relevant information surrounding the topic of the research question. For medical SLR's, this step typically involves developing a Boolean search query for a medical database such as PubMed that will return all relevant articles. Completing a full SLR is a very costly and time consuming task, frequently costing over $100,000 and taking 50+ weeks to complete and publish [2, 3]. Due to the extensive cost and time involved in completing an successful SLR, any strain that technology assisted reviews (TAR) can remove from the process while maintaining or improving overall performance is welcomed and encouraged.

The primary goal of this model is to improve Boolean search queries used in the study search step of SLR's using a neural approach to reduce cost, improve time efficiency, and improve the performance of manually developed search queries. We attempt to do this by producing a simple boolean search query using ChatGPT, and then adding relevant MeSH terms to the query based on the individual atomic terms in the original query. MeSH terms stands for Medical Subject Heading terms and are a controlled vocabulary organized hierarchically used for indexing and searching biomedical texts [4]. Research articles are often categorized with MeSH terms by the authors which assists in indexing, categorizing, and searching for said article. Due to the complexity of the MeSH term thesaurus, it is often considered a complex task to add relevant MeSH terms to a Boolean search query, however, having a strategic combination of both free-text and MeSH terms in a search query can improve the search results significantly [13]. This is the basis of the proposed model which will be further explained in the following sections.

II. Method

As previously mentioned, the proposed model attempts to improve SLR Boolean search queries by adding relevant MeSH terms based on the individual atomic terms. To do this, the model uses 3 pre-trained transformer-based models which are used in the three different steps of the model: ChatGPT Query Generation, BERTMeSH Term Suggestion, and finally Bert-Based Semantic Comparison.

A. ChatGPT Query Generation

First, our model needs to get a basic Boolean query that has the ability to be run on PubMeds database and will return articles relevant to the SLR. To do this, I implemented a method which involved prompting ChatGPT with the title of the SLR to return a Boolean query as proposed in [6]. Table I shows a sample SLR title, the associated ChatGPT query, and the resultant Boolean search query.

Once ChatGPT has returned a query, the query is split into query fragments by its AND operators, and then split into atomic terms by its OR operators. By splitting the query up this way, it allows us to consider the query by its individual free-text terms to determine which MeSH terms would be most relevant to each.

B. BERTMeSH Term Suggestion

This step of the model involves a pre-trained BERT-based model called BERTMeSH which was developed to index medical-related text with relevant MeSH terms. This model has the ability to take in a full medical text as an input and return a list of MeSH terms for that text [7][8].

The first thing we do before using the BERTMeSH model is iterate through each atomic term and feed each free-text

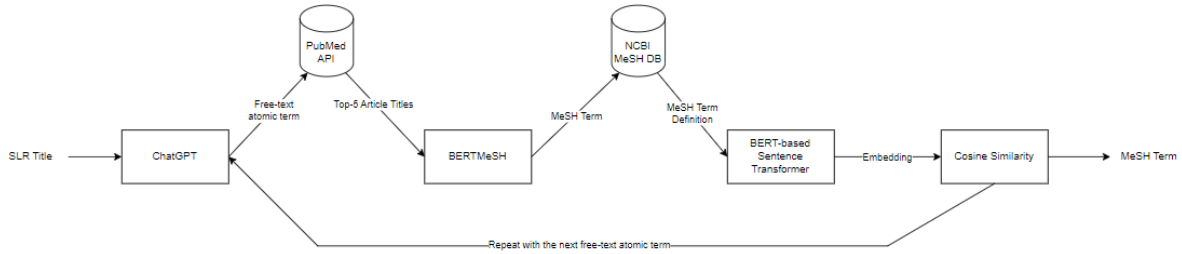


Fig. 1: Front-to-Back model description including three different pre-trained models and the two databases/API's accessed

SLR Title	Computed tomography angiography or magnetic resonance angiography for detection of intracranial vascular malformations in patients with intracerebral haemorrhage
ChatGPT Prompt	Based on the following SLR title, please provide 5 complex pubmed Entrez formatted query without descriptions, in plain text, such that they may be used directly on Pubmed's website. Please do not include any MeSH terms: Computed tomography angiography or magnetic resonance angiography for detection of intracranial vascular malformations in patients with intracerebral haemorrhage
Query	("Computed tomography angiography" OR "CT angiography") AND ("intracranial vascular malformations" OR "intracranial vascular abnormalities") AND ("intracerebral haemorrhage" OR "intracerebral bleeding")

TABLE I: Example of an SLR title from the CLEF TAR 2017 dataset, the ChatGPT prompt used, and the resultant query provided by ChatGPT

term into the PubMed API and get the titles of the first 5 resultant articles. From the success of the ChatGPT-Title method proposed in [6], it is inferred that a significant amount of semantic information can be extracted from the title of a research article alone. Further, we assume that the first 5 returned articles from PubMed after a free-text term is searched can be considered the most relevant articles to that free-text term.

The titles of the top 5 most relevant articles for each free-text atomic term is then fed into the BERTMeSH model which then returns a list of possible MeSH terms.

C. BERT-Based Sentence Transformers

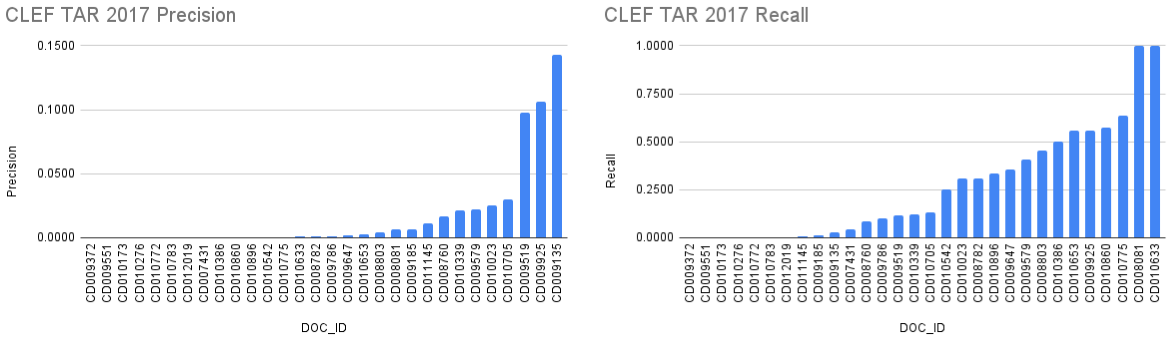
The final component of the proposed model involves using a BERT-based transformer model for semantic sentence comparison. The particular model used is outlined in [8, 9], and is used to create embeddings for two different sentences, then cosine similarity can be used to semantically compare the two embeddings and output a score between 0 and 1. The closer a score is to 1, the more semantically similar the two sentences are.

From the previous step, we have the query broken down into atomic terms, and we have a list of MeSH terms for each free-text atomic term. Next, take each MeSH term we found in the previous step and search them over NCBI's MeSH database which contains all MeSH terms as well as their summaries/definitions, and take the definition from the first search result for each term [10]. We then use the BERT-based Sentence-Transformer to embed each of these definitions and use cosine similarity to compare the definition embeddings with the free-text atomic term. Whichever MeSH term definition gives the best comparison result is selected as the most relevant MeSH term for that free-text atomic term and will be added into the original query, connected to the free-text term by an OR operator.

Once this process is done for each free-text atomic term, the query can be reformatted in the original format with the atomic terms in a fragment connected with OR operator, and the fragments connected with AND operators.

D. Justification

There were many different options when it came to the design of the proposed model. First of all, the idea to use ChatGPT to develop the initial query was ultimately decided on due to the tight time constraint of the project, and the level of expertise needed to implement an alternative method. One alternative method that was considered early on in the design process was to use the manually generated queries included in the testing dataset as a starting block for the query. After looking through the CLEF TAR 2017 and 2018 datasets, it was discovered that many of the queries provided were not runnable on the PubMed database and would need to be translated into PubMed syntax. Doing this would both require significant manual effort and expert knowledge. Additionally, a lot of the manual queries included in the datasets follow complicated formats whereas ChatGPT will typically produce a query composed of atomic terms connected with OR operators to form query fragments, and query fragments connected with AND operators. While splitting up the variably formatted queries would be possible, due to the time constraint it was decided that using ChatGPT would be more accessible.



(a) (b)

Fig. 2: Sorted precision and recall of each individual query from the CLEF TAR 2017 dataset

Next, the BERTMeSH model was an easy decision to include in the overall design of the proposed model. BERTMeSH was originally designed to be used towards indexing research grants with proper MeSH terms, however, I decided to use it for the application of indexing relevant documents for Boolean search queries [9]. Alternative designs for this section of the model could have involved fine tuning a pre-trained GPT model to generate relevant MeSH terms, however as MeSH terms need to use the exact spelling and representation as in the MeSH database, a model specifically trained on MeSH terms was more appropriate.

Finally, with regards to the semantic comparison step, there were a few alternatives that were considered before deciding on using BERT-based sentence transformers to compare the free-text term with the MeSH definitions. First of all, as all of the MeSH terms returned in the previous BERTMeSH step were all somewhat relevant to the free-text term, all of them could have been included in the query. However, I decided that this would make the query too long and possibly include too many unrelated resultant articles. An alternative method that I would like to explore in the future would be to use a BERT-based model trained or fine tuned on medical data rather than general knowledge sentences. The BERT-based sentence transformer I used was trained on sentences including "That is a happy person" and "Today is a sunny day", however the sentences we were embedding were along the lines of "Acute kidney failure resulting from destruction of EPITHELIAL CELLS of the KIDNEY TUBULES. It is commonly attributed to exposure to toxic agents or renal ISCHEMIA following severe TRAUMA"[9][11]. It is likely that a model trained on medical data would have been better equipped to embed the MeSH definitions appropriately, however, due to the time constraint and lack of available training data, I decided to move forward with the generic sentence transformer model.

III. Implementation

This model was programmed in and run using Google Colab. A pro membership may be required due to the size of the BERTMeSH model. Code blocks must be run in sequential order from the top down in the python notebook including the code block with the imports at the very top of the notebook. Note that transformers 4.22 is the required version of transformers, any newer versions will not run the BERTMeSH model. Before running the final block of code, you must make sure you have a folder named "results" which is where the text files containing the resultant article IDs will be exported to, and a folder named "titles" where the topic files from the datasets will be stored. Below I will give a brief description of what each code block/function is responsible for.

The first two code blocks are used for imports and for loading the BERTMeSH model to use later. The next code block with the function search_pubmed() is used to query pubmed with the final query and return the resultant article IDs. Next is the code block containing the pubmed_data() function which takes a PMID and queries pubmed, to get the titles of the top 5 returned articles. This is used when searching the atomic terms through the database to get the most relevant documents. Next is the code block containing the get_completion() function which is used to prompt ChatGPT and returns the ChatGPT generated result (you may need to input your own api_key). Next is the code block that contains the query splitting functions split_query_to_frags() and split_query_to_terms(), the first of which takes in a full Boolean query, splits it by its AND operators and stores it in a list, and the second of which takes the fragments made in the previous function, splits them by their OR operators, and stores the atomic terms in another list. The next code block contains the functions used to get the MeSH term definitions. The first function is called get_mesh_id() which takes a mesh term, searches the MeSH database, and returns the id associated with the term. The second function is called get_mesh_definition() which takes a MeSH term, calls the get_mesh_id() function, then uses the MeSH id to get the MeSH term definition. The next code block loads the sentence transformer model and has the function semantic_similarity() which takes in a free-text term and a list of MeSH term definitions, embeds each of them, and then uses cosine similarity to find the most relevant definition. The next two code blocks are used to import from the topic files in the CLEF TAR databases and export the pubmed IDs to the results file. The importing is done by iterating through each file in the "titles" folder, getting the title

from it, and adding it to a list of SLR titles. The exporting is done by formatting the PubMed API URL with the final query, creating a file in the "results" folder named with the PMID of the SLR, querying PubMed, and writing the resultant IDs to the file. The final code block has multiple nested loops which together go one by one through each of the topics in the CLEF TAR dataset and calls the various functions to get a resultant query and the IDs after using it to search over PubMed.

IV. Evaluation

The proposed model was run over both the CLEF TAR 2017 and CLEF TAR 2018 datasets to measure its performance. The metrics used to measure the performance includes precision, recall (including both recall @100 and recall @1000), and f-measure (including F0.5, F1, and F3). To run the model code over the CLEF TAR datasets, I downloaded the topic files from the testing folders and stored them in a folder called "titles" in the same directory as the ipython notebook.

After running the model over the CLEF TAR 2017 and 2018 datasets, a few observations can be made. First of all, there were quite a few queries that retrieved none of the relevant articles which resulted in both a precision and recall value of 0. This can be visualized well in Figure 2 which shows that specifically, over the CLEF TAR 2017 dataset, 7 out of the 30 queries returned no relevant documents, and over the CLEF TAR 2018 dataset, 13 out of the 60 queries returned no relevant documents. This means that the model completely fails its task on 22.2% of its queries, which is a concerning metric regardless of how the other queries perform.

A. Smooth Operator Comparison

The primary objective of this project was to create a model which operates within the same domain and tackles the same challenge as an existing model, and compare their results. The model we will be comparing these results to is the Smooth Operator Model shown in [12]. The Smooth Operator Model uses a probabilistic approach to improve SLR search queries by replacing Boolean operators with smooth operator equivalents. The metrics resulting from running the Smooth Operator Model over both the CLEF TAR 2017 and 2018 datasets are shown in tables II and III where the "Boolean Operator" performance is the standard performance using the translated queries provided in the datasets, "Smooth Operator Equivalent" shows the performance of the queries after the Boolean operators were replaced with their smooth operator equivalents, "Predictor" shows the performance of a model used to predict the Theta values in the smooth operators, and "Oracle", the best performing method, shows the performance of the smooth operators with their optimized Theta values. The performance of the model outlined in this paper run over the same two datasets is denoted as "MeSH Model" and can also be seen in tables II and III.

You can see in tables II and III that the overall recall, recall @100, and recall @1000 was significantly lower with my proposed model than with any of the other methods. In the Smooth Operator Model, it was determined that the worst performance came from the predictor method, and as we can see, my proposed model results in a recall 48.55% lower than the predictor method over the CLEF TAR 2017 dataset, and 51.61% lower than the predictor method over the CLEF TAR 2018 dataset. The best method proposed in the Smooth Operator paper was the Oracle method, which has a 192.91% better recall over the CLEF TAR 2017 dataset and a 182.71% better recall over the CLEF TAR 2018 dataset than my proposed model. While these values show that my proposed model is significantly worse than the existing method, the other metrics used to measure performance showed more comparable results. Over the CLEF TAR 2017 dataset my model has a 83.52% better precision, 61.48% better F0.5 score, 8.73% better F1 score, and a 75.1% better F3 score than the best performing method from the Smooth Operator Paper. Over the CLEF TAR 2018 dataset, my model has a 104.74% better precision, 57% better F0.5 score, 5.79% better F1 score, and a 35.93% better F3 score than the best performing method from the Smooth Operator Paper. My model having a relatively low Recall while having a relatively high precision and F-measure is interesting because it means that a decent number of the papers retrieved by the queries are considered relevant, however, out of the total number of relevant document, a very small number of them were retrieved by the search queries.

V. Discussion

As previously discussed in the evaluation section, my proposed model performs better in some metrics, namely precision and f-measure, and worse in other metrics than the baseline method. I believe it's important to understand that my model and the baseline model are completely different approaches to try and tackle the topic of improving SLR Boolean search queries. The baseline model takes a probabilistic approach and tries to change the function of the operators used in the queries but doesn't manipulate the actual content of the queries themselves. My model doesn't try to change the operators (other than the initial query generation through ChatGPT) and instead tries to add terms and change the content of the query. Based on the result, both approaches clearly have merit and should be further explored as avenues through which SLR Boolean search queries can be improved. There may even be merit in combining these two approaches by both adding MeSH terms to the original query and implementing smooth operators.

Some areas that I believe improvement could be made on my model and where further research is needed includes the forming of the initial query, the semantic comparison model, and the implementation of the MeSH terms into the original query.

First of all, the use of ChatGPT to form the original query was used due to its results in this unpublished research article [5], but depending on the application, I believe results could improve by translating the original manually made queries and using them instead. This method was not used in this model due to the time constraint, however especially when considering the recall values, I believe great improvement could come from using the original manually created queries.

Next, I believe the BERT-based sentence transformer model that I used to semantically compare the definitions of the

TABLE II: The performance of The Smooth Operator model and my proposed model in terms of recall, precision, and f-measure over the CLEF TAR 2017 dataset

	Recall	R@100	R@1000	Precision	F0.5	F1	F3
Boolean Operators	0.7521			0.0157	0.0214	0.0264	0.0429
Smooth Operator Equivalents	0.7521	0.2033	0.4747	0.0157	0.0214	0.0264	0.0429
Predictor	0.5106	0.1461	0.3259	0.0107	0.0121	0.0137	0.0198
Oracle	0.7695	0.2167	0.4812	0.0091	0.0122	0.0149	0.0245
MeSH Model	0.2627	0.0403	0.1390	0.0167	0.0197	0.0162	0.0429

TABLE III: The performance of The Smooth Operator model and my proposed model in terms of recall, precision, and f-measure over the CLEF TAR 2018 dataset

	Recall	R@100	R@1000	Precision	F0.5	F1	F3
Boolean Operators	0.8344			0.0204	0.0297	0.0385	0.0699
Smooth Operator Equivalents	0.8344	0.1807	0.5367	0.0204	0.0297	0.0385	0.0699
Predictor	0.6205	0.1464	0.3544	0.0206	0.0293	0.0372	0.0637
Oracle	0.8487	0.1923	0.5375	0.0211	0.0307	0.0397	0.0718
MeSH Model	0.3002	0.0680	0.2123	0.0432	0.0482	0.0420	0.0976

MeSH terms to the free-text term could be improved for this implementation by training or fine-tuning it on biomedical texts. As previously discussed, this sentence transformer model was trained on generic sentences not specific to any knowledge domain, and therefore it may produce better results in this domain to train it on biomedical texts in the future. Additionally, it may help the semantic comparison if the model also obtained the definition of the free-text term to compare it to the definition of the MeSH terms. In my implementation I only compared the embedded term not the definition, and obtaining the definition may make the comparison more accurate.

Finally, the last area that I would like to improve on, or do more research into would be the optimal number of MeSH terms that would be needed to improve the query the most. In the paper shown in [13] they also took the approach of improving SLR Boolean search queries by adding MeSH terms, and they used multiple different methods of deciding how many MeSH terms to add to the final query. The approach that I adopted involved only selecting one MeSH term for every free-text term, however, I found this often overcomplicated the queries. For example, there were a few free-text terms that didn't have a well matched MeSH term to associate with it, but with the method I implemented, one always needed to be added. I think this may be one of the reasons the recall was so far from the baseline, because an irrelevant MeSH term was added where no MeSH term was needed in the first place. In the future, I think there would be merit in adopting a predictive model that could determine if a MeSH term would improve the results or worsen them.

REFERENCES

[1] N. Jahan, S. Naveed, M. Zeshan, and M. A. Tahir, "How to conduct a systematic review: A narrative literature review," Cureus, 2016. doi:10.7759/cureus.864

[2] M. Michelson and K. Reuter, "The significant cost of systematic reviews and meta-analyses: A call for greater involvement of machine learning to assess the promise of clinical trials," Contemporary Clinical Trials Communications, vol. 16, p. 100443, 2019. doi:10.1016/j.conctc.2019.100443

[3] R. Borah, A. W. Brown, P. L. Capers, and K. A. Kaiser, "Analysis of the time and workers needed to conduct systematic reviews of medical interventions using data from the Prospero Registry," BMJ Open, vol. 7, no. 2, 2017. doi:10.1136/bmjopen-2016-012545

[4] "Medical subject headings - home page," U.S. National Library of Medicine, https://www.nlm.nih.gov/mesh/meshhome.html

[5] L. Budau and F. Ensan, "FASS-BSLR: Fully Automated Scholarly Search for Biomedical Systematic Literature Reviews," [Unpublished].

[6] R. You, Y. Liu, H. Mamitsuka, and S. Zhu, "Bertmesh: Deep contextual representation learning for large-scale high-performance mesh indexing with full text," Bioinformatics, vol. 37, no. 5, pp. 684–692, 2020. doi:10.1093/bioinformatics/btaa837

[7] "Wellcome/wellcomebertmesh" Wellcome/WellcomeBertMesh, https://huggingface.co/Wellcome/WellcomeBertMesh (accessed Nov. 24, 2023).

[8] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using Siamese Bert-Networks," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019. doi:10.18653/v1/d19-1410

[9] "Sentence-transformers/all-minilm-L6-V2" sentence-transformers/all-MiniLM-L6-v2, https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 (accessed Nov. 24, 2023).

[10] "Home - Mesh - NCBI," National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/mesh/ (accessed Nov. 24, 2023).

[11] "Kidney tubular necrosis, acute - mesh - NCBI," National Center for Biotechnology Information, https://www.ncbi.nlm.nih.gov/mesh/?term=acute%2Bkidney%2Btubular%2Bnecrosis (accessed Nov. 25, 2023).

[12] H. Scells, F. Schlatt, and M. Potthast, "Smooth operators for effective systematic review queries," Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023. doi:10.1145/3539618.3591768

[13] S. Wang, H. Scells, B. Koopman, and G. Zuccon, "Automated mesh term suggestion for effective query formulation in systematic reviews literature search," Intelligent Systems with Applications, vol. 16, p. 200141, 2022. doi:10.1016/j.iswa.2022.200141