# Information Retrieval: Matching Patient Cases to Clinical Trials - Part 3 - BERT*

Migla Miskinyte$^{1[0000-0003-1318-4809]}$

Universidade Nova Lisboa, Lisbon, Portugal

**Abstract.** The exponential growth and availability of data, ranging from numerical structured, semi-structured to complex unstructured data such as text, has increased the need for efficient natural language processing tools, including extraction of relevant information. In the first part of the project, we have implemented and validated two different information retrieval models, namely *Vector Space Model (VSM)* and *Language Model with Jelinek-Mercer Smoothing (LMJM)* in order to match patient cases to clinical trials. In the second part of the project, we further optimised VSM and LMJM models and using logistic regression as classifier for ranking, computed top 100 clinical trials using our improved retrieval algorithm. Third part of this project, was dedicated for understanding and implementing *Bidirectional Encoder Representations from Transformers (BERT)* contextual embedding model.

**Keywords:** Natural Language Processing · Information Retrieval · Language Model with Jelinek-Mercer Smoothing · Vector Space Model Logistic regression classification · Contextual Embeddings · BERT

## 1 Introduction

Processing and understanding natural language can be a tremendously difficult task that allows us to use technological advancements, for example for ad-hoc searches, passage retrieval, question answering or even for conversational search models [1]. Recently, with increased accessibility of patients medical records, multi-omics datasets and novel drug target models, information retrieval became an important part for development of precision medicine. One of the important tasks in the field is to be able to match patients and their symptoms to available clinical trials, for which a search engine has been already implemented [2].

Using the same clinical information dataset, composed of patient cases (queries) and corresponding relevant and non relevant clinical trials, this project was based on three different stages. **In the first part** of this project, we have learned about, implemented and validated two information retrieval models, namely *Vector Space Model (VSM)* and *Language Model with Jelinek-Mercer Smoothing (LMJM)*, for different n-grams in order to match patient information to relevant clinical trials. **In the second part** of this project, after creating index structure

* Supported by organization Universidade Nova de Lisboa.

for clinical trials, we have calculated relevance for each patient case based on different sections of our documents where we combined VSM and LMJM similarity scores as our predictors in order to train a logistic regression classifier to retrieve top 100 clinical trial documents. Finally, **in the last part** of this project we have studied clinically relevant BERT contextual embedding model by using one chosen query and chosen relevant and non relevant document. This allowed us to explore a complex BERT architecture, look at different BERT layers, attention weights for chosen words in different contexts.

## 2 Implemented methods

### 2.1 Vector space model

In order to extract the explicit keywords in the document, we can calculate *Tf-Idf* (Term frequency - inverse document frequency):

$$Tf - Idf(t, d_j) = \frac{f(t, d_j)}{size(d_j)}(\log \frac{||D||}{||d : t \in d||}) \tag{1}$$

where $||D||$ means the number of documents of corpus D; $||D : t \in d||$ is the number of documents containing term t, and size $size(d_j)$ the number of words of $d_j$. In order to retrieve and rank documents with regards to a specific query, after computing $Tf - Idf(t, d_j)$ vectors for the query and a document, we will compute the cosine similarity between both vectors. Cosine similarity is calculating similarity between two non-zero vectors of an inner product space:

$$sim(q, d_j) = cos(q, d_j) = \frac{\sum qt \cdot dj, t}{\sqrt{\sum q_t} \cdot \sqrt{\sum dj, t}} \tag{2}$$

### 2.2 Language Model with Jelinek-Mercer Smoothing

LMJM model sums probabilities of generating a term from a document with the probabilities of generating it from the collection using $\lambda$ smoothing parameter.

$$P(q|d) = \prod_{t \in q}((1 - \lambda)\frac{tf_{t,d}}{L_d} + \lambda\frac{tf_t}{L_c}) \tag{3}$$

where, $\lambda$ ranges from 0 to 1, $tf_{t,d}$ is a term frequency in our document, $tf_t$ is a term frequency in the corpus. $L_d$ is length of tokens in the document, $L_c$ is length of tokens in all of the documents.

### 2.3 LETOR model

LEarning TO Rank (LETOR) is a research area in the field of IR where machine learning models are used to rank a set of items. In this project, different sections of our documents (Brief Title, Brief Summary, Detailed description, Criteria) are being used as predictors to estimate how relevant clinical trial is for a patient case (a query). By using a logistic regression model, where our feature vectors are doc scores from LMJM and VSM models, corresponding weights from a training dataset are being used in order to rank the most relevant clinical trials from the test dataset (Fig. S4). Several other features or so called predictors or ranking signals can be used in a query-document ranking, for example scores resulting from other language models (TF, BM25), lengths and IDF sums of document's zones, document's PageRank, HITS ranks and their variants have been used benchmark LETOR datasets [3].

In Letor implementation we have been using a logistic regression classifier, which is a discriminative model for a binary classification. The logistic regression name comes from the logistic sigmoid function used in the predictive model.

$$\sigma(z) = \frac{1}{1 + e^{(-z)}} \tag{4}$$

where z is a linear function of several explanatory variables ($x_1$ to $x_n$) and regression coefficients ($w_1$ to $w_n$) and can be expressed as follows:

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n \tag{5}$$

### 2.4 Word embedding models

A word embedding is a learned representation of document vocabulary where words that have the same meaning have a similar representation. Using word embedding is used for capturing context of the word in a document, semantic and syntactic similarity or relation with other words. Word2vec [6], Glove [9] and fastText [10] word embeddings are context independent, such models output just one embedding for each word, combining all the different senses of the word into one vector. Limitations of such models are that they can not capture the meaning of each word in different surrounding contexts. On the other hand, more recent ELMo [7] and BERT [8] looks at the entire sentence as it assigns each word an embedding, which captures the context of the word in a that sentence. ELMo uses a deep, bi-directional Long short-term memory (LSTM) model based on recurrent neural network (RNN) to create word representations. BERT uses Transformer-attention based model with positional encodings to represent word positions. While both of the latter models are contextual, ELMo is a character based model (inputs are characters rather then words) allowing meaningful computations of out of vocabulary words. However, recently it has been demonstrated that character based models are outperformed by subword based models (like BERT) for large corpus size [11].

### 2.5 BERT

Bidirectional Encoder Representations from Transformers (BERT) was built with two ideas in mind, the transformer architecture and unsupervised pre-training. The transformer architecture has a fully attention-based approach, that means, the attention function is described by using a query and a key-value pair to an output. The output is the weighted sum of the values, the higher the weight assigned to the value, greater is the compatibility function of the query with the corresponding key. It also has an unsupervised pre-training structure, meaning that its weights are learned in advance through masked language modelling and next sentence prediction. The idea of unsupervised pre-training structure has the purpose of predicting the missing word given the left and right context of multiple phrases (bidirectionality), whereas the next sentence prediction objective is to understand whether one sentence follows the another.

For BERT to check which keys weight more for each query, attention uses a compatibility function, which assigns a score to each pair of words (as aforementioned) and then makes the dot product of the query vector with the key vector. Because the score can have negative results it is normalised using the softmax function. When applying the attention to the word embeddings we have produced the composite embedding and the greater the result of the composite embedding is, the greater relation there is between the query and the key. We can visualise these weights with head-view from layer 0 to 23 (in our case or in case BERT Base 0 to 11), which each operates on the output of the layer that came before. BERT has multiple heads (attention mechanisms) which operate in parallel to one another yielding dimensional output values. These values are then concatenated and again projected Layers x Heads (in our case 24x16).

In the context of clinical data, there are several **clinical BERT models** available, such as:

– BioBERT, which is initialized with the general BERT model and in pre-trained using PubMED and PMC articles.
– ClinicalBERT, pre-trained on 2M clinical notes in the MIMIC-III database.
– SciBERT, pre-trained on a random sample of 1.14 M full-text articles from Semantic Scholar (most of them biomedical papers, although some computer science related).
– Biomedical-slot-filling, which is initialized with BioBERT based on a slot filling approach that surpasses the need for entity and relation-specific training data.

## 3 Experimental setup

### 3.1 Datasets

Our main corpus, which corresponds to clinical trials and our queries can be downloaded from: RI-2021-Clinical Trials. In order to optimise our algorithms, we have decided to limit our corpus to the entry of "brief summary", which in total sums to 3625 entries. In the next part of the project, after optimisation steps, we intend to use the total corpus with positional indexing and divide our dataset into training and testing in order to provide an unbiased evaluation of our final model. The queries that are being used in our models are long, 5 to 10 sentence patient case descriptions that simulates an admission statement in an Electronic Health Record (EHR).

### 3.2 Data pre-processing for LETOR

In a second part of the project, we have created different sections for our corpus, consisting of text fields, such as Brief Title, Brief Summary, Detailed description, Criteria, as well as some non-text fields, such as gender, minimal and maximal age. First, we have noticed that we had several fields (attributes in xml file) that were missing all together from some of the documents, resulting in different lengths of our sections (for example NCT02006251 docID had a different xml format from the rest of the documents). In order to keep our sections same length, if attribute did not exist, for Detailed Description, Brief Summary we have added a Brief Title and for one file that had a Criteria attribute missing, we decided to add as "Undisclosed".For the attributes that were present, but were empty, we added a text from Brief Title for that document ID. For missing gender fields, we added "both", missing minimal age, we added "0 Years" and for missing maximum age, we added "100 Years".

## 4 Results and discussion

### 4.1 LMJM and VSM implementation

We have implemented VSM model with different n-grams (unigrams, bigrams and trigrams). After computing previously described metrics, we can clearly see that this model is mostly adequate for unigrams with sharp decrease in metrics performance in bigrams and trigrams (Table 1). Recall was 0.983 in all cases, because in our model we retrieved all relevant documents by the search. This metric will be more relevant when we split our dataset in training and testing in next part of the project. Similarly, MRR metric will only be used in the next part of the project if we will rank correctness of possible responses to the samples of queries.

When implementing LMJM model for unigrams with $\lambda$ smoothing parameter ranging from 0.1 to 0.9 (Table 2), we have discovered that 0.8 $\lambda$ increased AP, P@10 and NDCG5 very close to VSM model (see also Fig. S1). The best value of $\lambda$ depends on the specific collection and query. In our case, our parameter gives more smoothing and we are increasing the importance of the collection model, and diminishing the importance of document model.

Moreover, when looking at metrics for individual queries we have observed that in many cases when P@10 for a query is large than average precision is increased as well (Fig. S1). Moreover, when looking at metrics variation, we can see that P@10 and NDCG5 form two clusters, one below and another above mean average (see Fig. S3).

### 4.2 LETOR implementation

In a second part of the project, we have further optimised VSM and LMJM models in order to calculate similarity scores between our queries and different fields of our corpus, such as Brief Title, Brief Summary, Detailed description

Table 1: Summary of average metrics for VSM

| n-gram | P@10 | Recall | AP | NDCG5 | MRR |
|---|---|---|---|---|---|
| Unigram | 0.068 | 0.983 | 0.043 | 0.062 | 0.005 |
| Bigram | 0.035 | 0.983 | 0.022 | 0.045 | 0.005 |
| Trigram | 0.008 | 0.983 | 0.010 | 0.016 | 0.005 |

Table 2: Summary of average metrics for LMJM (unigrams)

| $\lambda$ | P@10 | Recall | AP | NDCG5 | MRR |
|---|---|---|---|---|---|
| 0.1 | 0.048 | 0.983 | 0.033 | 0.039 | 0.005 |
| 0.2 | 0.048 | 0.983 | 0.035 | 0.044 | 0.005 |
| 0.3 | 0.051 | 0.983 | 0.036 | 0.046 | 0.005 |
| 0.4 | 0.056 | 0.983 | 0.037 | 0.049 | 0.005 |
| 0.5 | 0.055 | 0.983 | 0.040 | 0.052 | 0.005 |
| 0.6 | 0.060 | 0.983 | 0.041 | 0.055 | 0.005 |
| 0.7 | 0.060 | 0.983 | 0.042 | 0.061 | 0.005 |
| 0.8 | 0.060 | 0.983 | 0.042 | 0.062 | 0.005 |
| 0.9 | 0.063 | 0.983 | 0.042 | 0.054 | 0.005 |

and Criteria. The summary metrics for each of the predictors varied sightly between fields for unigrams for LMJM and VSM models (summaries of **per field and model retrieval performance** in Table 3). Scores were used as feature vectors in order to train a logistic regression classifier, where we have split a query-document pairs into a training and testing datasets (at 80/20 proportion). **Feature Scaling**. Before training our classifier, we have scaled our data using a z-score normalization (normalizing every feature in a dataset such that the mean of all of the features is 0 and the standard deviation is 1). Additionally, we have also tried range normalization, but this did not improve performance of our classifier. **Model regularization**. In order to avoid overfitting, we used a cross validation in order to estimate the best regularization parameter C. When looking at our dataset, we have noticed a significant **Class Imbalance**, with a ratio of 0.71 for 0 not relevant classes and 0.29 for relevant classes. Hence, for our logistic regression model we have balanced class weights using a built-in function.

**Model metrics**. After training our logistic regression with a training dataset (all 8 features), we have obtained an accuracy of 0.52, and an accuracy of 0.50 with our test set. This very slight difference is not unexpected with our model very slightly overfitting. While our classification accuracy is low, it can be misleading metric as accuracy measure alone is not a very reliable measure of model performance, when datasets do not have equal class distribution. Indeed, when we look at other metrics, such as precision and F1 score, we can see that our model performs better with larger class (N for non relevant cases, Table 4). From our metrics it is clear that we have a high rate of false positives and false negatives in our confusion matrix. To increase overall model accuracy and precision

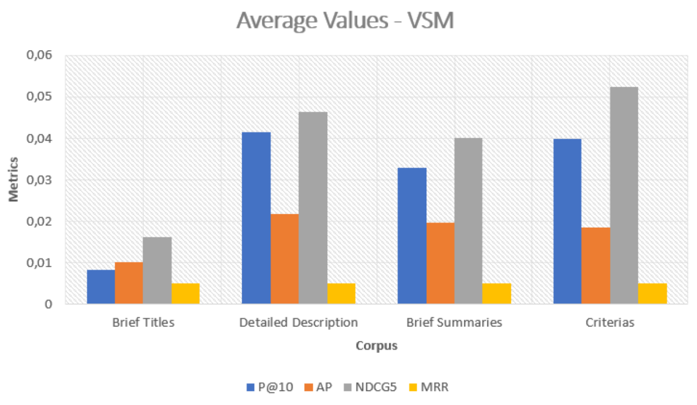


Fig. 1: Comparisation of the average values of the metrics P@10, AP, NDCG@5 and MRR for different fields for LMJM model.

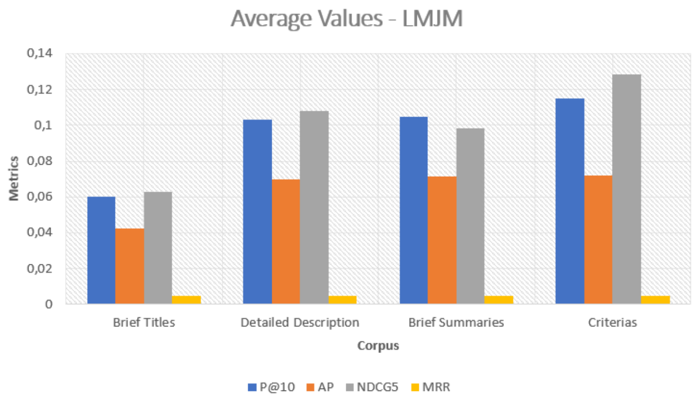


Fig. 2: Comparisation of the average values of the metrics P@10, AP, NDCG@5 and MRR for different fields for VSM model.

Table 3: Summary of average metrics for different fields for VSM and LMJM models (unigrams)

| **VSM** | **Field** | **P@10** | **Recall** | **AP** | **NDCG5** | **MRR** |
|---|---|---|---|---|---|---|
| | Brief Titles | 0.0083 | 0.9833 | 0.0103 | 0.0163 | 0.0050 |
| | Detailed Descriptions | 0.0416 | 0.9833 | 0.0217 | 0.0463 | 0.0050 |
| | Brief Summaries | 0.033 | 0.9833 | 0.0198 | 0.0402 | 0.0050 |
| | Criterias | 0.040 | 0.9833 | 0.0185 | 0.0524 | 0.0050 |
| **LMJM** | **Field** | **P@10** | **Recall** | **AP** | **NDCG5** | **MRR** |
| | Brief Titles | 0.0600 | 0.9833 | 0.0424 | 0.0627 | 0.0050 |
| | Detailed Descriptions | 0.1033 | 0.9833 | 0.0696 | 0.1077 | 0.0050 |
| | Brief Summaries | 0.105 | 0.9833 | 0.0714 | 0.0983 | 0.0050 |
| | Criterias | 0.1149 | 0.9833 | 0.0721 | 0.1286 | 0.0050 |

Table 4: Classification report for Logistic regression model

| **All data before criteria adjustment (accuracy = 0.52)** | | | | |
|---|---|---|---|---|
| Relevance | Precision | Recall | F1-score | Support |
| N | 0.72 | 0.53 | 0.61 | 2764 |
| R | 0.3 | 0.5 | 0.37 | 1106 |
| **All data after criteria adjustment (accuracy = 0.64)** | | | | |
| Relevance | Precision | Recall | F1-score | Support |
| N | 0.76 | 0.71 | 0.74 | 2764 |
| R | 0.38 | 0.44 | 0.41 | 1106 |

and recall, we further adjusted a model by filtering out many false positives. This was done by filtering out other criteria (gender, minimum and maximum ages) for our patient descriptions that do not match a particular clinical trial.
After training and perfecting our logistic regression classifier, we ranked our most relevant queries and document combination using a score which is calculated as a linear combination of all features (all similarity scores) weighted by the parameter values (logistic regression coefficients). When looking at importance of each of field with high impacts for individual query (sorted by the max absolute value) and different retrieval models in Figure 3 using SHAP (SHapley Additive exPlanations [4]) method that can be used to explain individual predictions, we can see that features f6 and f4 have highest importance for individual query, however very slightly so. Estimated Shapley values (x-axis), which corresponds to a contribution of a feature value to the difference between actual prediction and the mean prediction given all the set of features, are actually very similar. This is also obvious when looking at weights $w_1$ to $w_8$ of our logistic regression model (Table S1).
**Fig.1 and Fig.2 Observation:**An examination on the relationship between the collection frequencies (LMJM) and the document frequencies (VSM) of the terms in several representative TREC collections indicates that they have both strong and significant correlations. As we may verify by comparing the two graphs (Fig.1 and Fig.2), each metric is quite similar in terms of its behaviour.

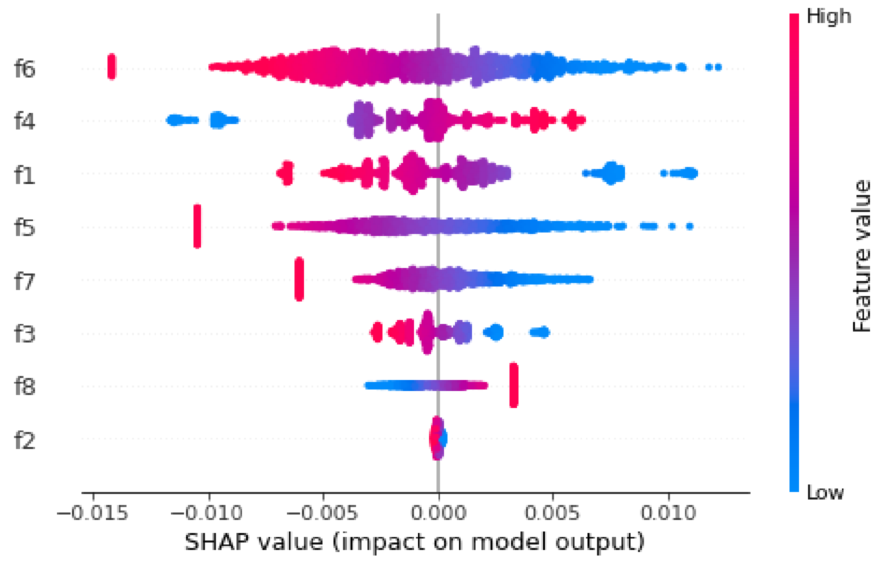


Fig. 3: Overview of feature importance for a Logistic Regression classifier using SHAP values. Where features are as follows: f1 - Detailed Descriptions (LMJM), f2 - Brief summaries (LMJM), f3 - Criteria (LMJM), f4 - Brief Titles (LMJM), f5 - Detailed Descriptions (VSM), f6 - Brief summaries (VSM), f7 - Criteria (VSM), f8 - Brief Titles (VSM).

## 4.3 BERT implementation

There are several different pre-trained BERT models that could be used for NLP tasks in a clinical context (several were described in Chapter **2.5 BERT**). For this project we have chosen to implement a novel biomedical BERT model biomed-slot-filling instead of other also relevant Biomedical BERT models. It has an advantage over other models as it is based on a slot filling approach, replacing the need for entity and relation-specific training data. This is very important in querying biomedical data, where queries can contain multiple relevant answers [12].

We have chosen a query 'A 44-year-old man complains of severe headache and fever. Nuchal rigidity was found on physical examination.' (query id ['201512'] - sentence A) and a relevant document - 'volunteers who need a myelogram of their spine as part of their routine medical care are being asked to be in this study.' ('NCT00231374' - sentence B) based on highest ranking in LETOR model and based on our text understanding. We have chosen a random non relevant trial for the BERT implementation ('NCT00097838' - sentence C). Our inputs in the model are pairs of query and documents, either sentence A and sentence B, or sentence A and sentence C. Unexpectedly during tokenization process, some words, such as "nuchal" into subwords: 'n','##uch', '##al' or "rigidity" into rigid','##ity' were separated into different tokens, indicating that those words are not commonly found in a corpus. Our tokenized input sequence of chosen length, has 3 different representations, that are summed element wise to produce an input representation, which is passed to BERT's encoder layer:

- token embeddings representation (with a shape of (1, 1,1024), which are vector representations of words.
- segment embeddings with the same shape and 2 vector representations (indexed as 0 and 1 for different passed pairs).
- position embeddings allowing BERT to learn a vector position for each representation.

In other words, each produced token has its own unique index, which is later transformed so we have a unique vector that represents it (word embeddings). During tokenization, model specific tokens are also added, to indicate the start of the sentence ([CLS]) and where it separates and ends ([SEP]), the tokens are encoded into their corresponding IDs and attention masks are added (which explicitly differentiate real tokens from [PAD] tokens). In the next part, our input representation is passed to BERT's encoder layer. Our chosen BERT model has 24 layers (plus our input layer), 1 batch, 55 tokens (as we truncated) and 1024 features. In order to visualise conceptually each token representation in different BERT stages, we have extracted 10 chosen token embeddings from the first layer (our input) and the last layer from Sentence A and Sentence B pair (Fig. 4) as well as same tokens from the pair of Sentence A and Sentence C (Fig. S5). Looking at both Figures, we can see that the first layer is static (they do not change), while the last layer is dynamic (same word position is slightly different between in Sentence A and Sentence B pair input and sentence A and Sentence

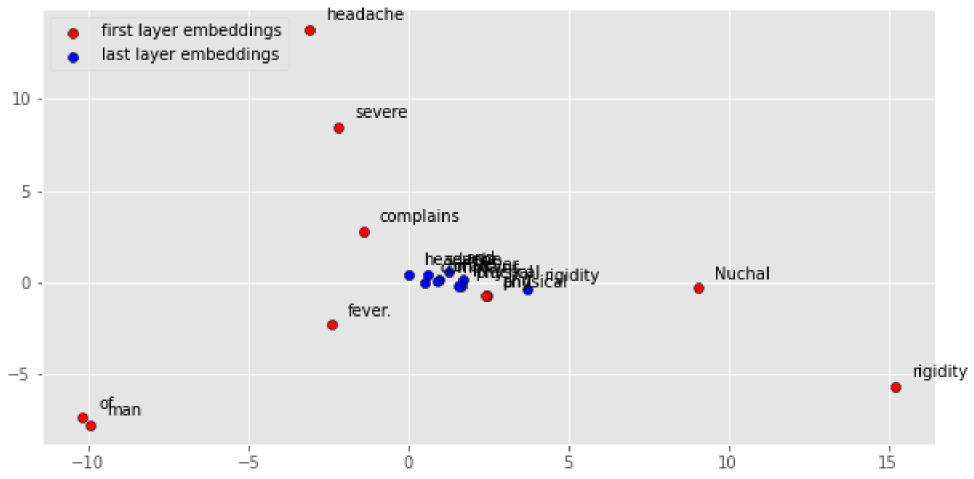
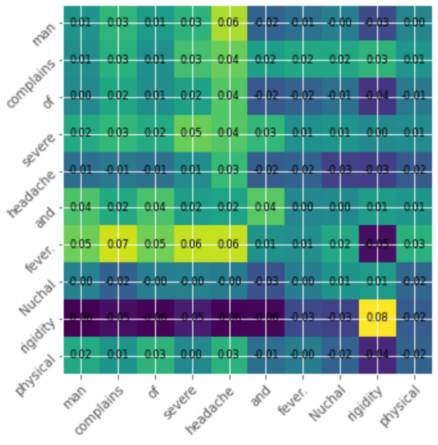


Fig. 4: Chosen tokens in a sentence (query and relevant document) using first and last vectors embeddings.

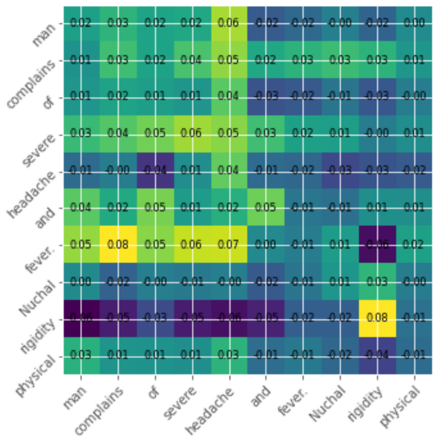
C pair input). Moreover, we can see that our tokens are grouped together, they look very comparable in the last layer vector embedding.

Another way of visualising how different tokens are represented in the first embedding layer and in the last embedding layer is computing a similarity measure between our tokens using a cosine similarity between our representative vectors. As expected, when we compare our chosen words in different layers we can see that representation of same words in different layers is very different (e.g. ”man” in first layer vs ”man” in last layer is 0.01 and 0.02 in Figure 5 a) and b) respectively), token embedding vectors are orthogonal to each other, they do not match (Fig. 5). We have also demonstrated how the same chosen tokens are represented in the same BERT layer (in this case, last layer) between different inputs (Sentence A and Sentence B vs Sentence A and Sentence C). Because BERT produces different vector embeddings based on the surrounding context, it is not surprising that similarity score between same words is not equal to 1 (e.g. see Fig. 5 c) ”rigidity” =0.95). As expected, similarity scores for same tokens (”rigidity” and ”rigidity”) is 1 in the static embedding layer. There were two unexpected findings when looking at Fig. 5 c), first, word ”rigidity” is very dissimilar to other tokens in the context, second, the word ”nuchal” that was split into 3 different subwords during tokenization, has a high similarity score to other words, however closest to ”fever”.

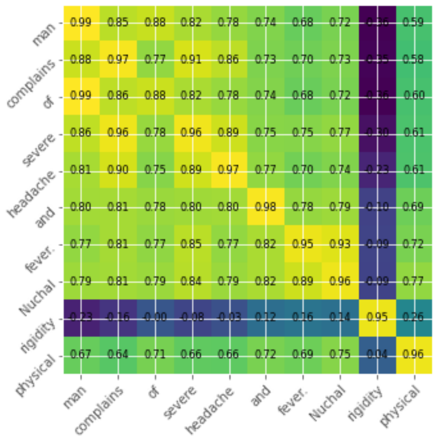
Next, we have looked at attention weights in different BERT levels using two different approaches. In the first approach, we have visualised attention weights from a full model (where our inputs where 55 tokens from pairs of sentences) and



(a) Similarities between first and last layer embeddings when using query and relevant docID for model input



(b) Similarities between first and last layer embeddings when using query and non relevant docID for model input



(c) Similarities between last layer embeddings of query and relevant docID vs. query and non relevant docID

Fig. 5: Similarities between chosen tokens at different embedding layers and different contexts (when same tokens are chosen from different input sentences)

in the next approach, we have visualised attention weights from a model where we have passed as input only our tokens (10 chosen words, resulting into 15 subword tokens). We did this in order to demonstrate how attention weights for the same word are recalculated given different input. The more words that are present in total in each sentence, the more ambiguous our chosen word in focus becomes. As expected, attention weight is stronger for the same word in a shorter input. Moreover, we have noticed that in layer 0 attention weights are stronger with itself (see example for "severe" - "severe", especially in head 0, Fig. 4.3, this is a representation of the model that had only 15 subwords). If we look at attention weight for token "severe" in a layer 0 using head visualisation, we can see that "severe is paying attention to the word that was in vicinity "headache". Overall, there some BERT's attention heads exhibit strange patterns such as attending to [SEP] and [CLS] tokens or broadly attending over the whole sentence, with heads in the same layer often showing very similar behaviors.

## 5 Conclusions

After implementing VSM and LMJM models, we have concluded that that VSM performs slightly better for unigrams when we compare mean average percision. However, average precision decreased for VSM with increase in n-grams. While, we could not run 2-grams or 3-grams for LMJM (because of lack of memory), we would expect that it would outperform VSM for longer queries. In a second part of the project, we have further optimised VSM and LMJM models in order to calculate similarity scores between our queries and different fields of our corpus, and using them as predictors (feature vectors) we have trained a logistic regression classifier. Furthermore, using weights obtained from logistic regression model, we have implemented LETOR model and obtained top 100 clinical trials. Given the current set of feature values, the contribution of each feature value to the difference between the actual prediction and the mean prediction is very similar. However, we could expect that each contribution could change slightly every time we shuffle and train our model. Moreover, it would be interesting to estimate how our model would change for bigger n-grams.

In the third part of the project, we have explored contextual word representation model BERT, which recently revolutionized the field of NLP. It has been very powerful for different tasks, for sequence to sequence based language generation tasks, such as question answering, sentence prediction, conversational response generation, and also natural language understanding tasks as polysemy, word sense disambiguation, sentiment classification and others. Moreover, recently voice based text recognition systems started being used with BERT. In our project we have just started understanding an architecture of this complex model and it would be very interesting to dive deeper into it. For example, how would our results change if we process batch paired sentences? Would pooling last 4 output layers of BERT produce best results as proposed for some tasks in the original paper [8] or this is more task dependent? How would different clinical BERT models change our results?

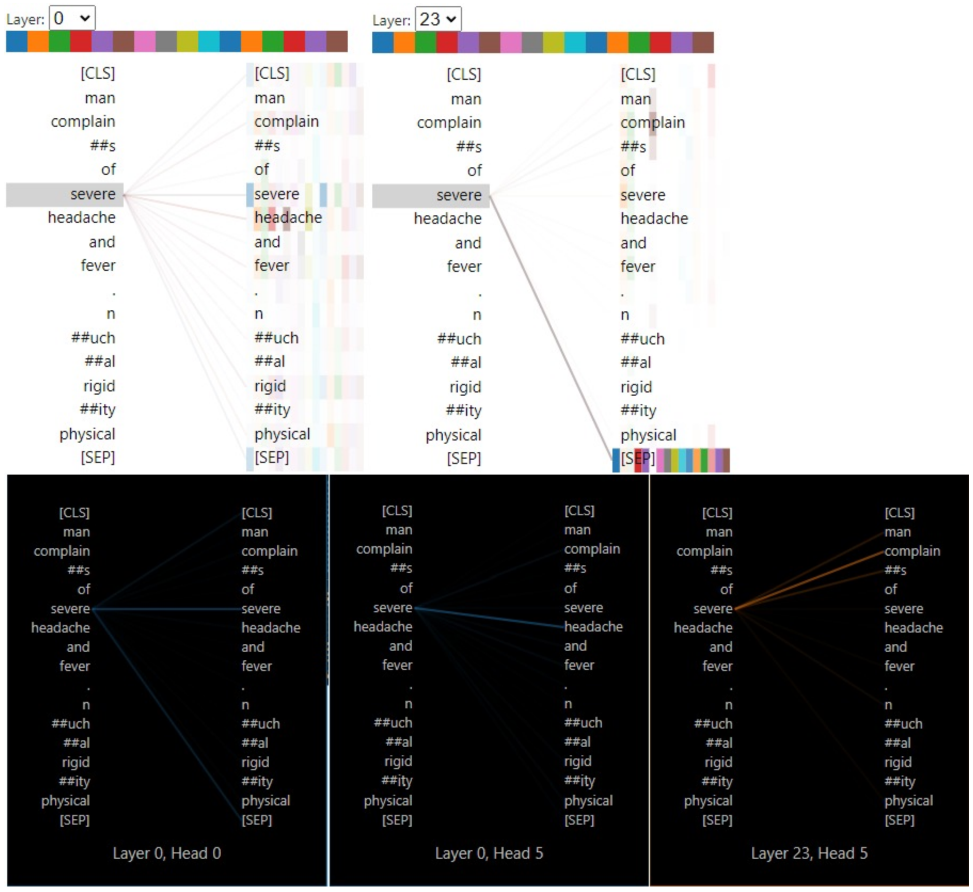


Fig. 6: **Layer 0 all attention:** The above visualisation shows one attention mechanism within the model, meaning in layer 0 we have a higher weight in key "headache" to query "severe" and it might be informative head.**Layer 24 all attention:** Most attention is focused on the [SEP] token, this pattern does not serve much for anything, it means that an attention head can not find anything else in the input sentence to focus on. **Layer 1 head 0:** Because there is only one mechanism used in the layer, there are not many representations and therefore the key with the word "severe" has a higher weight for the query "severe". **Layer 1 head 5:** In the above figure it illustrates a different weight distribution. This happens because there is a different head. As a consequence, different learned linear projections are used resulting in a higher weight of the key "headache" in detriment to the weight of "severe". **Layer 23 head 5:** At 23 layer and with the same attention parameter as before, the key with the higher weight for the query "severe" is "complain". Resulting in 24x5 distinct attention mechanisms used.

## References

1. Radlinski, F. and Craswell, N: A Theoretical Framework for Conversational Search. Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval **10**,pp. 117–126 (2017)
2. LNCS Homepage, `https://medical.novasearch.org/`. Last accessed 3 Nov 2021
3. Qin, T., Liu, T.-Y, Li H: LETOR: A benchmark collection for research on learning to rank for information retrieval. in *Inf. Retr.*, pp. 346–374, (2010)
4. Lundberg, Scott M., and Su-In Lee.: A unified approach to interpreting model predictions. in *Advances in Neural Information Processing Systems* (2017)
5. C. D. Manning, P. Raghavan and H. Schütze: Introduction to Information Retrieval. 3rd ed. draft. Cambridge University Press, 2008
6. Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems. 2013.
7. Matthew E. Peters, et al. "Deep contextualized word representations." Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 2018.
8. Ashish Vaswani, et al. "Attention Is All You Need ." 31st Conference on Neural Information Processing Systems. 2017.
9. Pennington, JeffreyP et al. "GloVe: Global Vectors for Word Representation." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.
10. Young,Julio Christian and Rusli, Andre. "Review and Visualization of Facebook's FastText Pretrained Word Vector Model".2019 International Conference on Engineering, Science, and Industrial Applications (ICESI). 2019.
11. Al-Rfou,Rami et al. "Character-Level Language Modeling with Deeper Self-Attention." The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19). 2019.
12. Papanikolaou,Y and Bennett, F. "Slot Filling for Biomedical Information Extraction." arXiv.org ¿ cs ¿ arXiv:2109.08564v1 (under review). 2021

## 6 Supplementary Information

### 6.1 Metrics used in this paper

In order to evaluate our models, we have used different metrics for system utility, such as precision@10, nDCG5, recall, MRR, and stability, such as MAP [5].

Most commonly used metrics in model evaluation are accuracy, precision, recall and F1 score. For, which we have used only precision and recall in our project:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$precision = \frac{TP}{TP + FP} \tag{7}$$

$$recall = \frac{TP}{TP + FN} \tag{8}$$

where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives are derived from a confusion matrix. In our case, we were given a dataset with relevance scores, which greatly facilitate implementation of the metrics.

Another useful metric in information retrieval is precision@k. This precision metric takes the number of queries that were identified correctly by the algorithm and divides by the total number of selected documents. For any given dataset, the more false positives the algorithm identifies (wrongly chosen documents), the lower the precision will be. The precision can be evaluated at given cut-off rank, considering only the topmost results returned by the algorithm. This measure is called P@n, where n is the number of results we want.

We also used the Recall@n. This recall metric, takes the number of queries that were identified correctly by the algorithm and divides by the number of relevant documents, which in our case every document is considered relevant. Because our documents are filtered, so that only the relevant ones get to be considered, therefore the recall will be theoretically 1. The recall can too be evaluated at given cutoff rank, considering only the topmost results returned by the algorithm. The measure is named R@n.

Example of Precision and Recall for document retrievals:

| **Documents** | **Precision** | **Recall** |
|---|---|---|
| Incorrect | 0 | 0 |
| Correct | 1/2 | 1/3 |
| Incorrect | 1/3 | 1/3 |
| Correct | 2/4 | 2/3 |
| Incorrect | 2/5 | 2/3 |

A widely used measure of ranking quality is the Normalized Discounted Cumulative Gain (NDCG), which measures the usefulness, or gain, of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks. For us to understand what the normalized DCG makes, first we need to understand why is it used instead of DCG (Discounted Cumulative Gain). The idea of DCG is to give more relevance to documents that are found in the beginning stages of our search and ever less so for documents later in the search. This is useful because for example, in order to take into account the tendency for older documents to get outdated, we may wish to grade them differently and we do so by sorting 1 them by date. The other example is the reputation of where the information comes from. In principle anything that could indicate a higher or lower quality of information could be used in the sorting metrics. The normalized DCG, will simply normalize the values of the graded relevance used in the DCG.

$$nDCG@5 = \frac{DCG@5}{IDCG@5} \tag{9}$$

where DCG@5 is given by the formula:

$$DCG@5 = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{log_2(i+1)} \tag{10}$$

where $rel_i$ is either 0 or 1, corresponding to the document i being relevant or not. IDCG is given by the formula:

$$IDCG@5 = \sum_{i=1}^{p} \frac{2^{|rel_p|} - 1}{log_2(i+1)} \tag{11}$$

where $REL_p$ represents the list of relevant documents (ordered by their relevance) in the corpus up to position p.

A widely used statistics method is the Mean Reciprocal Rank. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer: 1 for first place, 1/2 for second place, 1/3 for third place and so on. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q. In order to have an idea of what the average lowest rank of each query is, with respect to the documents it's to be applied to, we simply order the documents in terms of the relevance they have in relation to the query, sum the reciprocals of each index and then divide it by the number of query. With this kind of metrics we can evaluate basically the first positions, which in our case means knowing if the first documents are good for our query.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{12}$$

where $rank_i$ refers to the rank position of the first relevant document for the i-th query.

For system stability metrics, we use the Mean Average Precision.

$$MAP = \frac{\sum_{q=1}^{Q} AveP(q)}{|Q|} \tag{13}$$

The Mean Average Precision (MAP) for a set of queries is the mean of the average precision scores for each query.

### 6.2 Supplementary Tables and Figures

Table S1: Logistic regression coefficients

| | **f1** | **f2** | **f3** | **f4** | **f5** | **f6** | **f7** | **f8** |
|---|---|---|---|---|---|---|---|---|
| **coefs** | 0.0034 | -0.0001 | -0.0013 | 0.0042 | -0.0043 | -0.0046 | -0.0024 | 0.0020 |
| **odds** | 1.0034 | 0.9999 | 0.9987 | 1.0042 | 0.9957 | 0.9955 | 0.9976 | 1.0020 |

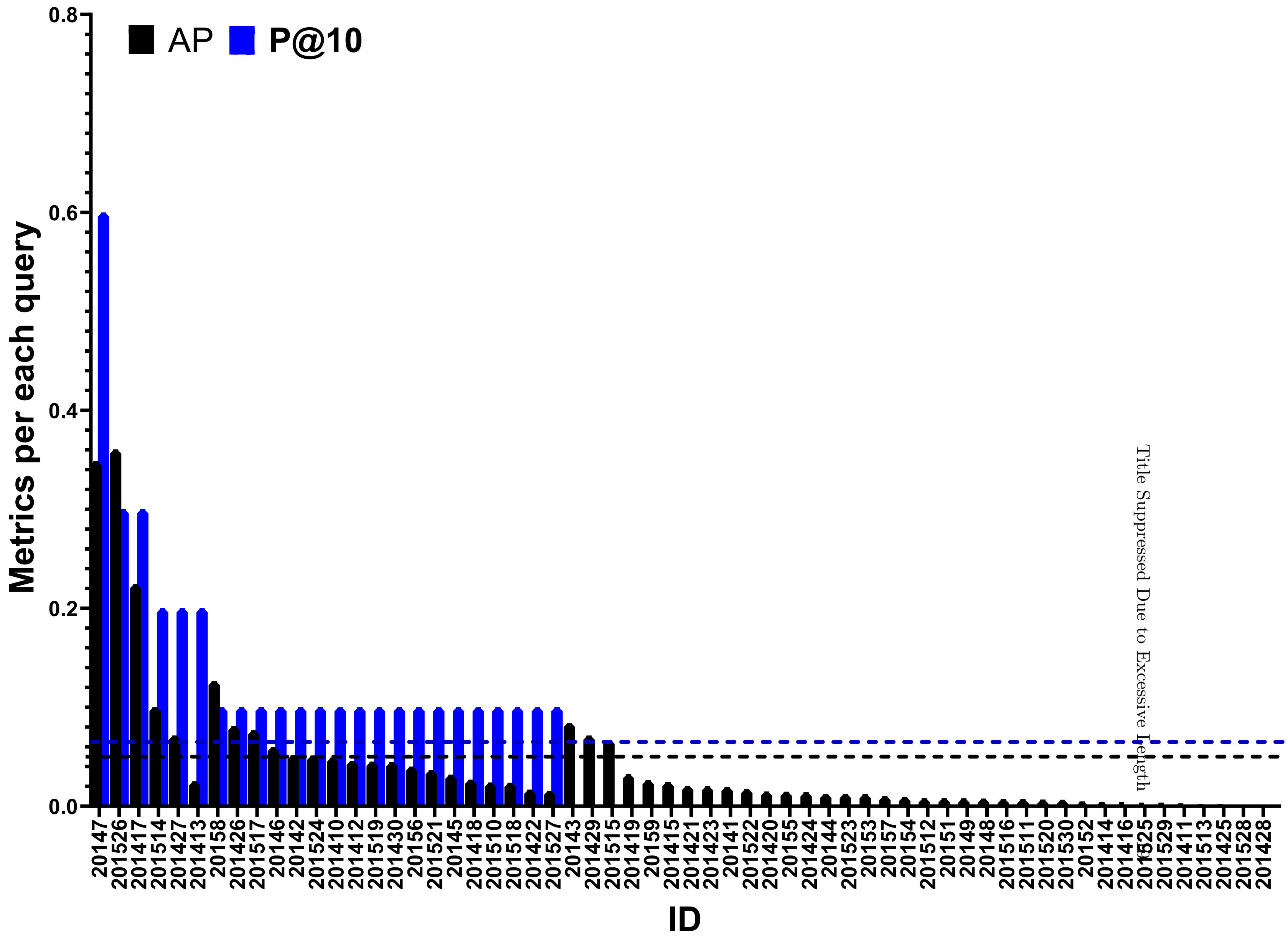


Fig. S1: Metrics per each query ID with 0.8 $\lambda$ smoothing parameters. Dashed black line indicates average AP and dashed blue line average P@10 for all queries.

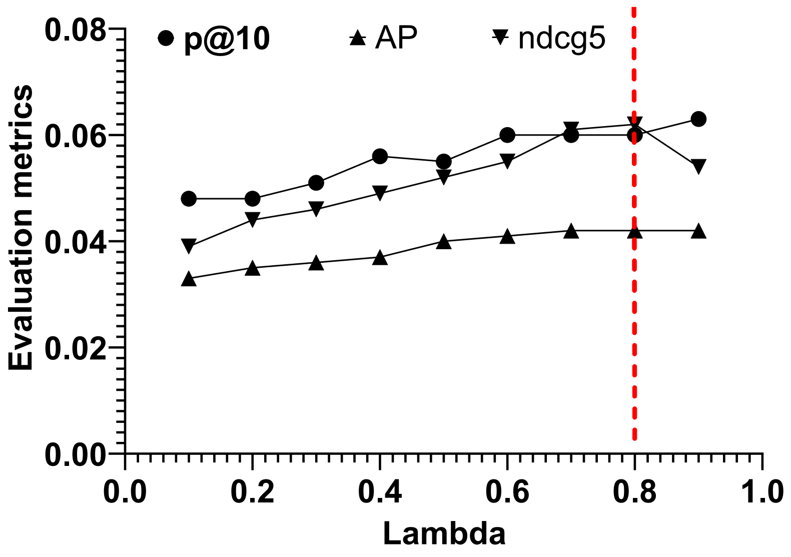


Fig. S2: LMJM model evaluation metrics using different $\lambda$ smoothing parameters. Optimal $\lambda$ was chosen at 0.8, which is indicated by a red dashed line.

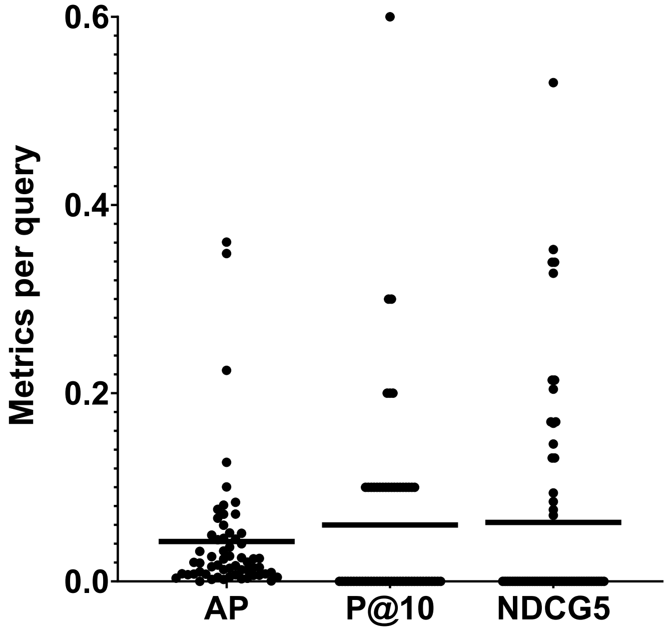


Fig. S3: Variation of metrics per query for LMJM model with 0.8 $\lambda$. Horizontal lines show mean for each metric.

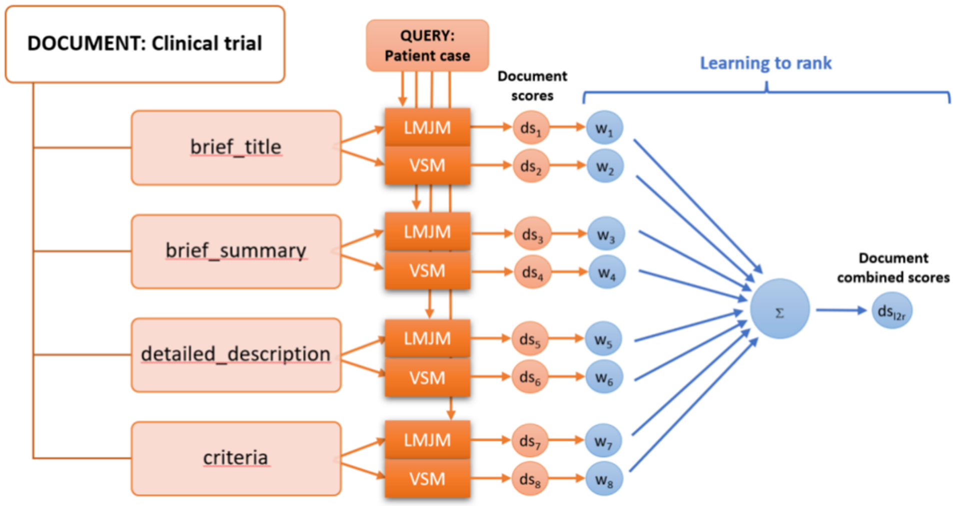


Fig. S4: Explanation of LETOR model. Adapted from Information Retrieval 2021 classes.

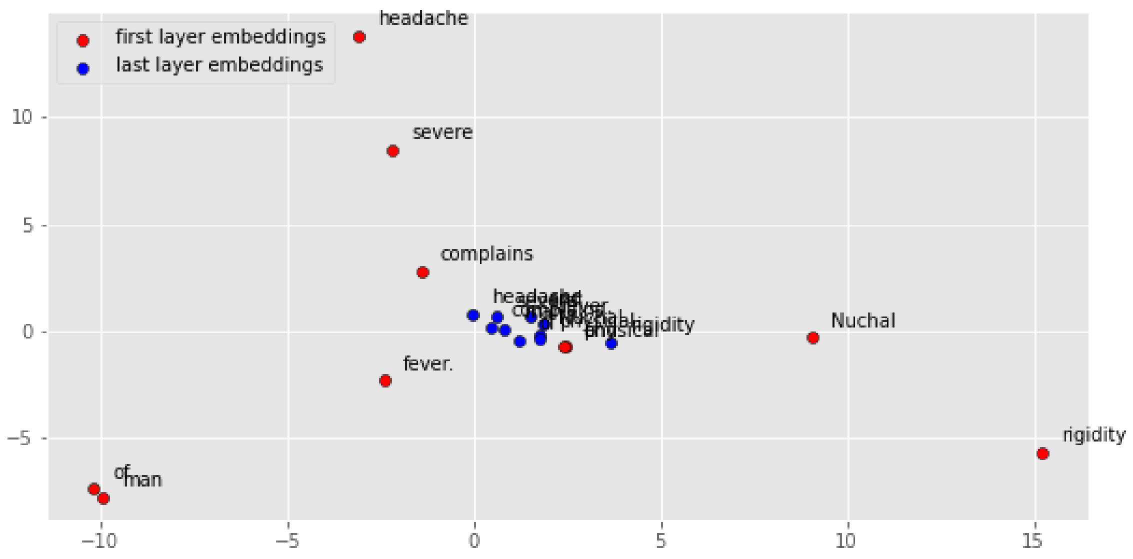


Fig. S5: Chosen tokens in a sentence (query and non relevant document) using first and last vectors embeddings.