Matching Patient Cases to Clinical Trials

Mateusz Małecki, Antoni Lasik and Mateusz Zieleziński

1. Introduction

1.1. Foreword

Modern medicine is facing a new branch of opportunities and new solutions that have never been thought of. Countries struggle with providing efficient healthcare to their inhabitants. Queues for medical diagnosis are absurdly long. Costs that are associated with maintaining such a system are enormous. But there is a way of diagnosing patients that is cheap, efficient, automated, and can be developed to be accurate. Our task was to mimic in some basic way the process of such a diagnosis.

1.2. Data set description

We were provided a dataset that consisted of multiple clinical trials, that were built of elements, such as ID, brief title, detailed description, summary, and some medical details. Also, we had a set of patient cases that consisted of the patient's id and description of the patient's condition. The third source of our knowledge was a file with pairs of patient cases and clinical trials and their relevance, previously labeled by assessors. As previously mentioned our task was simplified, build models that will find the most matching clinical trials to the patient case based on above mentioned resources.

2. Implemented methods

2.1. Data transformation

We have started with transforming the data into more suitable forms. The file with labeled patient cases and clinical trials had a spare column that we deleted (the second one) and we have fitted the data into the pandas DataFrame. From our database of clinical trials, we retrieved their ids and brief descriptions and we did put them into two lists: docs and ids. Patient cases were transformed into a dictionary. Later when we refer to documents we mean clinical trials brief descriptions and when we refer to queries we mean patient cases.

2.2. Data split

To make sure that our implementation is correct, we ran the test on whole dataset just to see if the values of the evaluation metrics will be the same as provided by the lecturer. Having the confirmation, that our model works as expected, it was time to perform the split of dataset. Like on every "learning" algorithm, we had to select hyperparameters in such a way, so the algorithms will perform possibly the best. To do so, we randomly selected 80% of queries and used them as a training set. The remaining 20% was used as a test set when our hyperparameters were already fixed.

2.3. Vector space model with term frequency–inverse document frequency weighting

Then we started creating the first model. We performed term frequency–inverse document frequency weighting using a function from sklearn, we have obtained a set of vectors where each vector is a representation of a document, and each value in this vector is a tf-idf weight of a term. Then the next step was to represent the query in the same way as the vector. We performed the same operation using sklearn and we did calculate the pairwise cosine distance and subtracted it from one to obtain the measure of similarity between a given query and the documents. We saved it as a score and put it in a DataFrame with the corresponding document ids, and ordered from the most similar to the least.

2.4. Language model with Jelineck – Mercer smoothing

The second model was a bit different. We started with computing the count matrix which was representing the occurrences of words in clinical trials titles. Each column in this matrix was a word i and each row represented a document j, so the entry Eij represented how many times word i occurred in document j. We used CountVectorizer for it and transformed the outcome into 2D numpy array. Then we constructed two additional matrices matrix_corpus and matrix_documents. Matrix_corpus is a 1D array when entry Ei was a possibility of occurrence of the word i in the whole corpus i.e. all of the documents joint together, this matrix was obtained by summing the columns of the initial matrix. Matrix_documents is a 2D matrix in which entry Eij represents the probability of occurrence of the word i in the document j. In the code it is done by the JMS function. Then we formatted the query in a way that it no longer had white spaces and new line characters at the end and later we used a function that transformed a query that it only consisted of words that occur in the corpus and we associated the words with their corresponding i indexes in the matrix_documents and matrix_corpus matrices. It is done by query_formatting and words_and_indexes_association functions. Later we calculated the scores with the formula provided by the professor, as a summation with a logarithm of two probabilities from matrix_documents and matrix_corpus with a lambda coefficient. We added these scores into the DataFrame and ordered from the most similar to the least.

3. Experimental setup

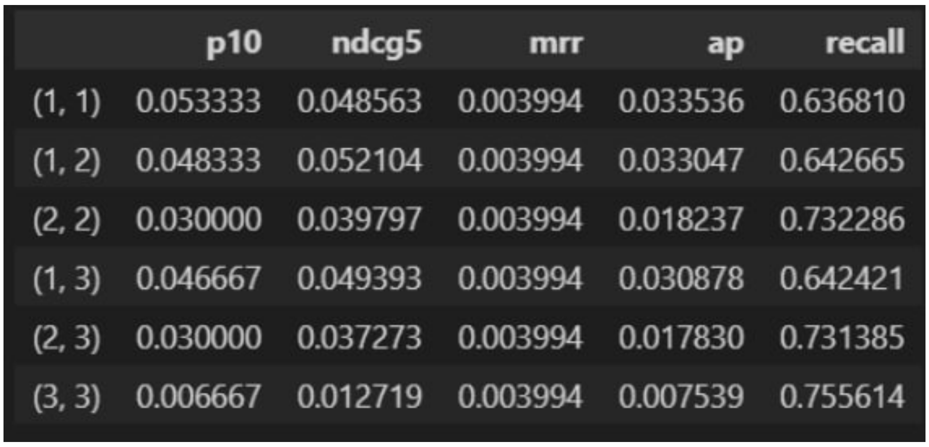
3.1. Metrics and setup

We created two functions that were evaluating the scores DataFrame, and were calculating some metrics. Metrics that we used are:

- Precision@10-precision at top 10 scores
- nDCG@5-Normalized Discounted Cumulative Gain in the top 5 scores.
- Recall@100-recall at top 100 scores
- Mean Average Precision
- Mean Reciprocal Rank

Then we have created a setup for two experiments, For tf-idf model we checked different ranges of ngrams and how the metrics change for each one of them. For the language model with Jelineck-Mercer smoothing we trained the model with different lambda values. In each experiment we were using training set of patient cases.

3.2. First experiment

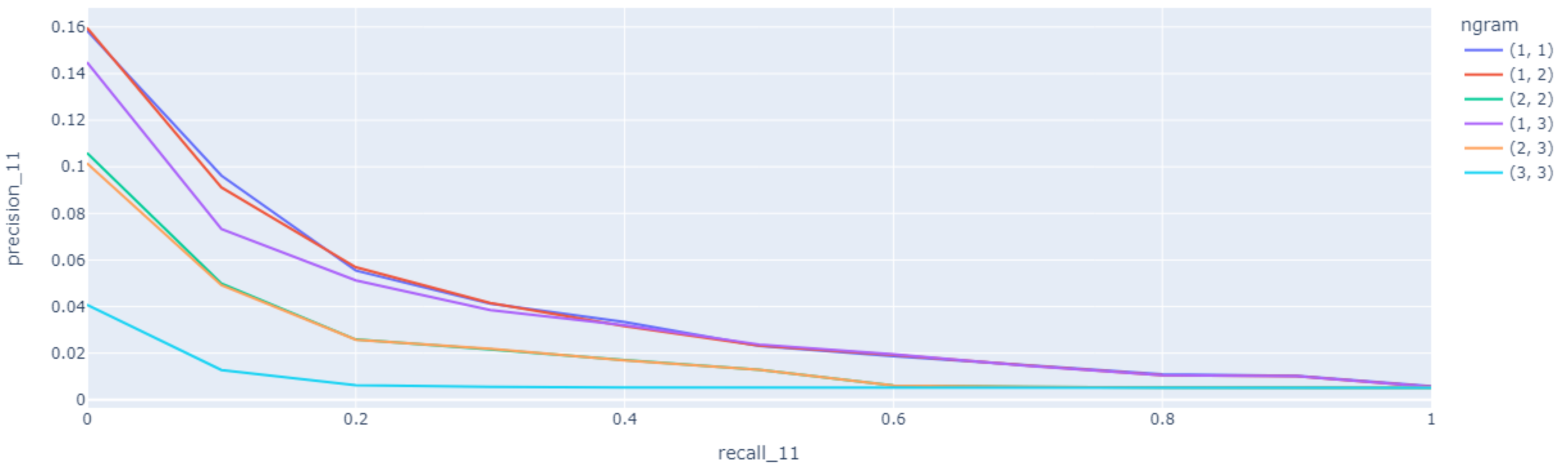


	p10	ndcg5	mrr	ap	recall
(1, 1)	0.053333	0.048563	0.003994	0.033536	0.636810
(1, 2)	0.048333	0.052104	0.003994	0.033047	0.642665
(2, 2)	0.030000	0.039797	0.003994	0.018237	0.732286
(1, 3)	0.046667	0.049393	0.003994	0.030878	0.642421
(2, 3)	0.030000	0.037273	0.003994	0.017830	0.731385
(3, 3)	0.006667	0.012719	0.003994	0.007539	0.755614

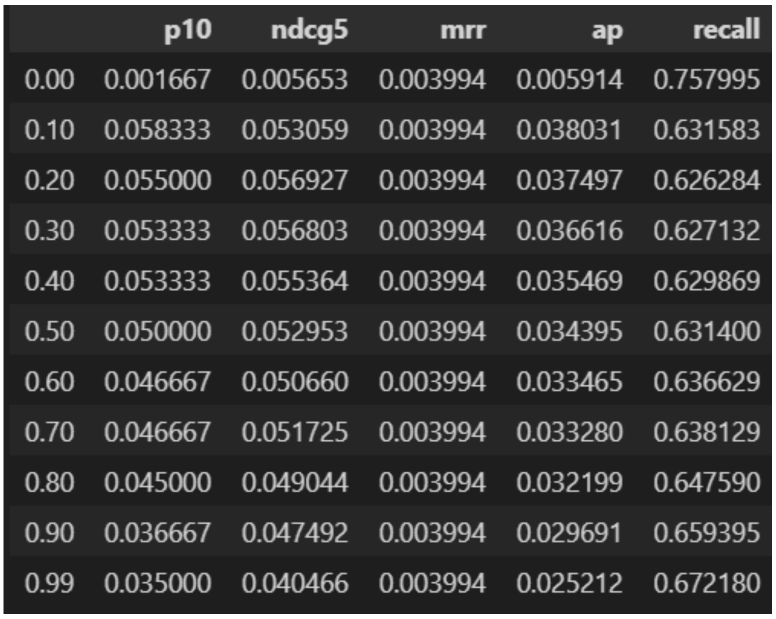
Fig. 1. For different sizes of n-gram range we acquired these values of metrics on training set.

We were focused on high values of precision@10 and ndcg@5, which were the highest in the n-gram=1, when the model was constructed using single words.

Fig. 2. We have also made a plot representing the interpolated precision-recall to check if (1,1) was the best choice.



3.3. The second experiment



	p10	ndcg5	mrr	ap	recall
0.00	0.001667	0.005653	0.003994	0.005914	0.757995
0.10	0.058333	0.053059	0.003994	0.038031	0.631583
0.20	0.055000	0.056927	0.003994	0.037497	0.626284
0.30	0.053333	0.056803	0.003994	0.036616	0.627132
0.40	0.053333	0.055364	0.003994	0.035469	0.629869
0.50	0.050000	0.052953	0.003994	0.034395	0.631400
0.60	0.046667	0.050660	0.003994	0.033465	0.636629
0.70	0.046667	0.051725	0.003994	0.033280	0.638129
0.80	0.045000	0.049044	0.003994	0.032199	0.647590
0.90	0.036667	0.047492	0.003994	0.029691	0.659395
0.99	0.035000	0.040466	0.003994	0.025212	0.672180

Fig. 3. For model with JM smoothing we started with list of 11 lambdas, and run evaluation for each one of them. These are the metric results.

As we can see the best results the model obtained around lambda equal to 0.2 or 0.1.

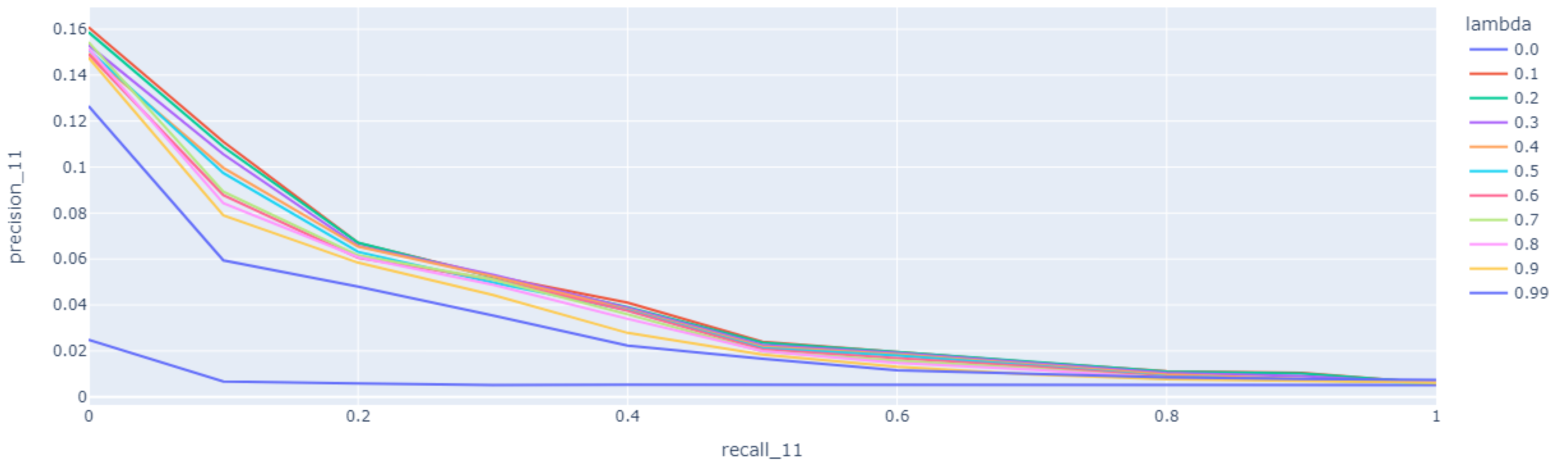
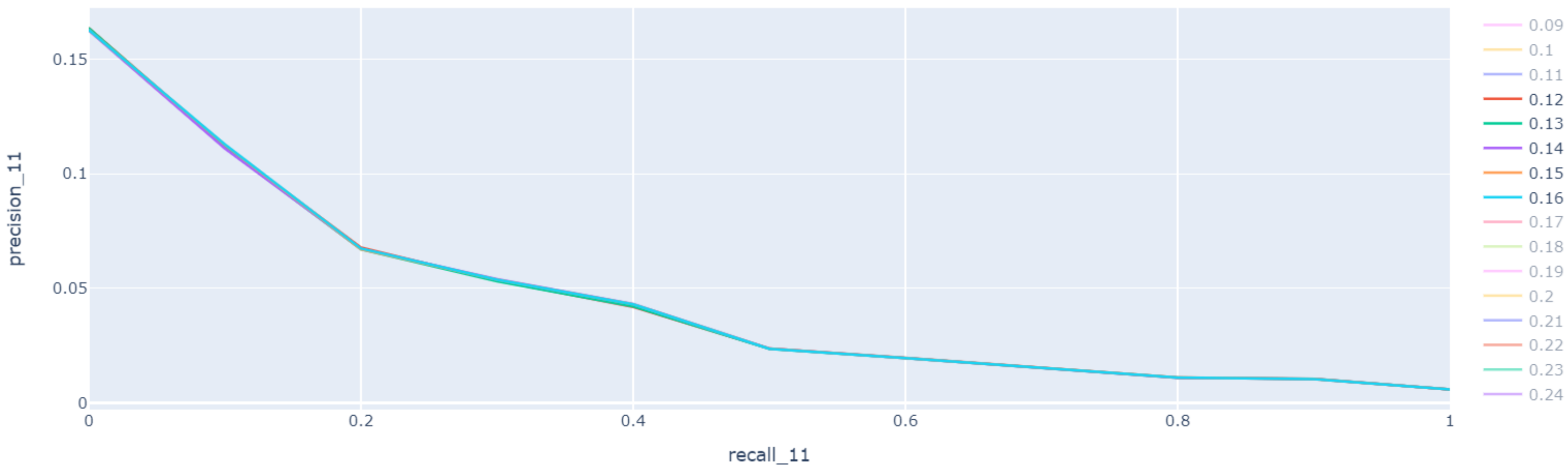


Fig. 4. Precision – recall curves for different lambdas

So we did another experiment for smaller range of lambdas around these values. The results were the best around lambda equal to 0.12.

Fig. 5. Precision-recall curve, the results were very similar.

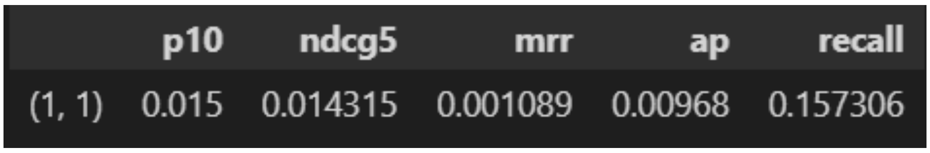


4. Result discussion

4.1. Test set run

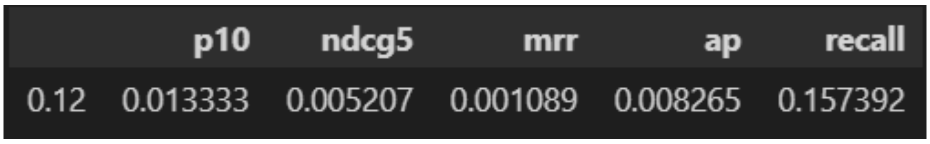
When we already had our parameters tuned, we ran evaluations on the test set.

Fig. 6. Results of VSM with td-idf for n-gram = 1



	p10	ndcg5	mrr	ap	recall
(1, 1)	0.015	0.014315	0.001089	0.00968	0.157306

Fig. 7. Results of LM with JMs for lambda=0.12.



	p10	ndcg5	mrr	ap	recall
0.12	0.013333	0.005207	0.001089	0.008265	0.157392

5. Conclusion

Looking at the results we don’t feel satisfied, there is a big difference between scores obtained on training set and test set. This might be the fault of overfitting, the parameters we chose for training data made model fitted only for this particular set, but on the other hand it could not be done in any more general way. The other case might be the lack of data processing. We didn’t delete any punctuation marks or didn’t perform lemmatization of queries or documents. Also changing everything to lower-case might have helped.

6. Phase two

In this phase we were supposed to use LETOR models based on VSM an LMJM and different fields of the document.

7. Data used and missing values

In the first phase of our project, we were using just the brief title field of clinical trials, in this phase we needed to retrieve also fields like brief summary, detailed description, and criteria. Some fields of the documents were empty so we had to fill them in, we used a combination of non-missing fields of the particular document. Also the criteria field had a different format, so we had to transform it into plain text and leave only the criteria that were inclusive.

8. Model adjustments

8.1 VSM

In this phase, we had to consider the different fields and tune the n-gram parameter for each field separately. So we created an experiment similar to phase one in which we but in this case, we just calculated the metrics for each field separately to know which value of n-gram parameter is the best.

8.2 LMJM

We expanded the experiment from phase one with the calculation of metrics for different fields on queries training set, we tried lambdas from 0 to 1 with step 0.05. We compared the results and selected the best-suited lambdas.

9. LETOR

9.1 Methods used in LETOR

matrix_x_creation

To create a LETOR model we had to create a matrix that stored for each labeled pair of query and document corresponding scores from VSM and LMJM used on different fields. Firstly we created a dictionary in which we stored a list of documents indexes labeled in pairs with that query for each index query. Then for each query in a set(training or test or validation), we calculated the score that this query obtained in a given model using four different fields(brief title, brief summary, detailed description, and criteria) and we added these scores as a row in a data frame, also we created an additional column a tuple(query index, document index) to serve as an id. Then we iterated through the matrix of scores to add to the final matrix only those that were labeled. For this operation, we used the tuple column.

matrix_y_def

To create a LETOR model we had to create a matrix that stored for each labeled pair of query and document corresponding scores from VSM and LMJM used on different fields. Firstly we created a dictionary in which we stored a list of documents indexes labeled in pairs with that query for each index query. Then for each query in a set(training or test or validation), we calculated the score that this query obtained in a given model using four different fields(brief title, brief summary, detailed description, and criteria) and we added these scores as a row in a data frame, also we created an additional column a tuple(query index, document index) to serve as an id. Then we iterated through the matrix of scores to add to the final matrix only those that were labeled. For this operation, we used the tuple column.

matrix_x_norm

For matrix x we performed normal min max normalization. We used MinMaxScaler from sklearn.preprocessing. We also dropped the tuple column to not cause problems in logistic regression

log_regr_train

In this function, we used Logistic regression with from sklearn.linear_model package. With input of matrices x and y. The function returned weights that the regression assigned to each column in matrix x.

calc_scores_with_coef

Is a function that based on coefficients(weights) and a query set calculates the document scores for each query and sorts them and then calculates the metrics based on these rankings.

10. Training the LETOR

10.1 Using the methods

Training the LETOR consisted of the functions above rightfully ordered and executed. The first step consisted of creating the matrix x and matrix y. Then using logistic regression and calculating the scores.

10.2 Data imbalance handling

We have trained logistic regression models with three different settings. In the first model, we called model A, here as matrix y we provided a vector of binary labels, either pair was relevant or no. In the second model-B, the logistic regression was trained with the parameter class_weight set to “balanced, so the weights per class are equal. The third model-C, we changed the sample weight parameter and used vector of relevances that had values from 0 to 2.

10.2 Regularisation parameter

For each model presented above, we tuned the regularization parameter c, which is the inversion of regularization strength, the lower the values of the, the stronger regularization we use. Tuning the regularization is used to prevent the model from overfitting or underfitting. To ensure that we don’t overfit with tuning the c parameter we used k-fold cross-validation with four folds on our query training set. For each value c from the list [0.1,0.3,0.5,0.7,0.9] we divided the set on training and validation sets. For the training set, we performed all the steps needed to train logistic regression, and the with obtained coefficients we ranked the documents for each query in validation set and calculated metrics then we summed all of the metrics from folds to calculate the average for given c parameter. We chose the best parameter c according to the highest value of the P@10 metric.

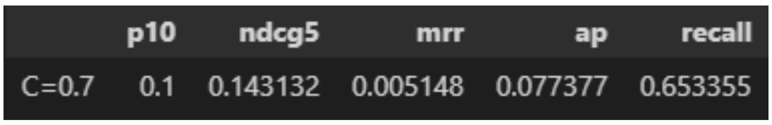
11. Evaluation and results comparison

Outcomes from first phase and the second phase may differ a bit because of the changed split.

11.1 Parameters

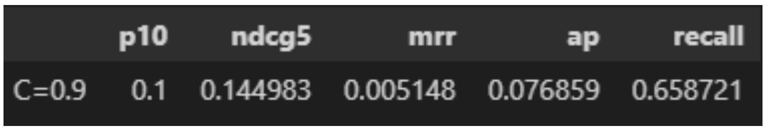
For all models we used the same n-gram range (1,1), For LMJM we have (0.10,0.25,0.25,0.15) for fields corresponding fields (docs_brief_title, docs_brief_-summary ,docs_detailed_description, docs_criteria). For LETOR model A we used C=0.7,for model B C=0.9 and for model C C=0.3.

Model A evaluation run



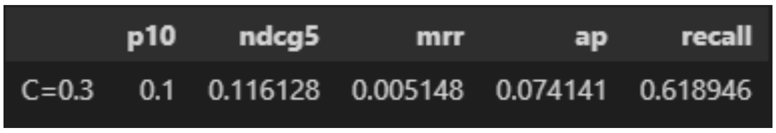
	p10	ndcg5	mrr	ap	recall
C=0.7	0.1	0.143132	0.005148	0.077377	0.653355

Model B evaluation run



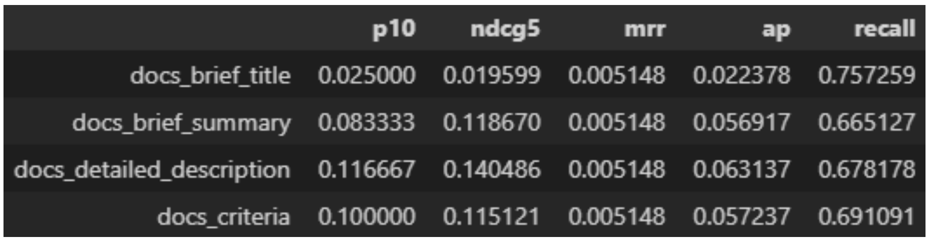
	p10	ndcg5	mrr	ap	recall
C=0.9	0.1	0.144983	0.005148	0.076859	0.658721

Model C evaluation run



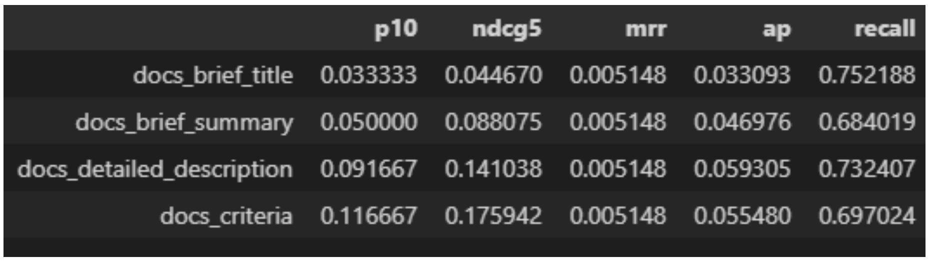
	p10	ndcg5	mrr	ap	recall
C=0.3	0.1	0.116128	0.005148	0.074141	0.618946

VSM evaluation run



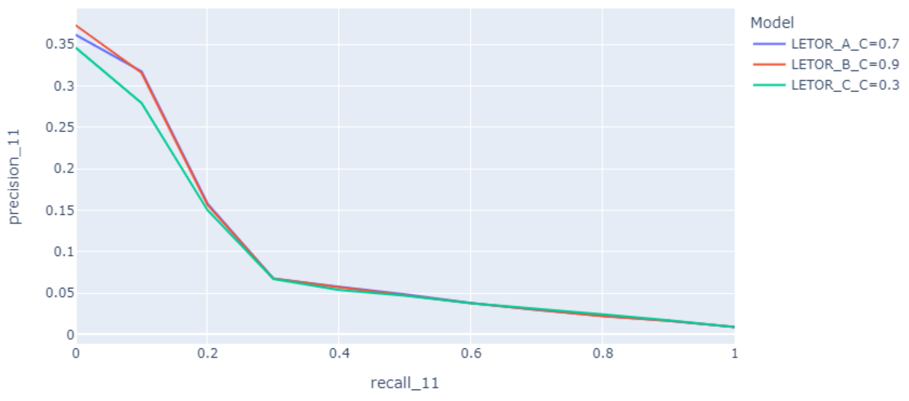
	p10	ndcg5	mrr	ap	recall
docs_brief_title	0.025000	0.019599	0.005148	0.022378	0.757259
docs_brief_summary	0.083333	0.118670	0.005148	0.056917	0.665127
docs_detailed_description	0.116667	0.140486	0.005148	0.063137	0.678178
docs_criteria	0.100000	0.115121	0.005148	0.057237	0.691091

LMJM evaluation run

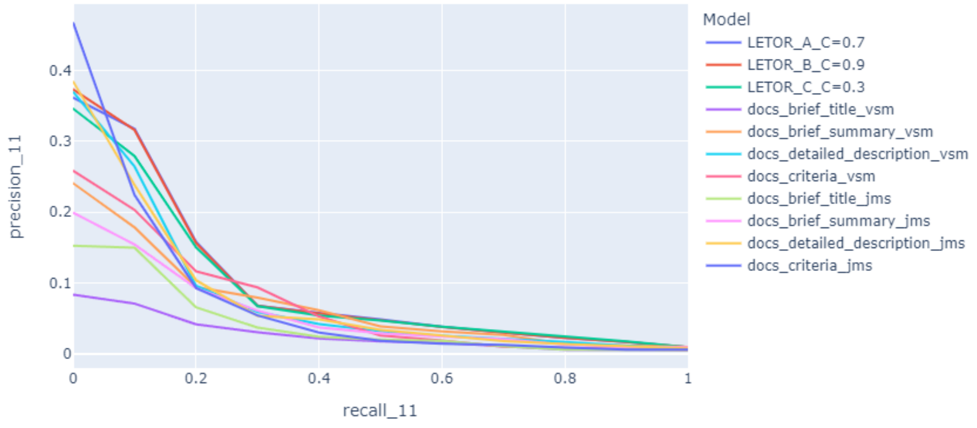


	p10	ndcg5	mrr	ap	recall
docs_brief_title	0.033333	0.044670	0.005148	0.033093	0.752188
docs_brief_summary	0.050000	0.088075	0.005148	0.046976	0.684019
docs_detailed_description	0.091667	0.141038	0.005148	0.059305	0.732407
docs_criteria	0.116667	0.175942	0.005148	0.055480	0.697024

LETOR models comparison



All models compared



12. Results discussion

The LETOR in all configurations is significantly better than VSM and LMJM based on single fields, as we can see the criteria field is most successful and obtained the highest precision in LMJM. Looking at the coefficients that logistic regression provided, lets take for example model A for a list:
[[-0.00548344 0.01690746 0.03142435 0.03545309 0.01936886 0.03388526 0.03758225 0.0252886]]
We can see that the first two values are the lowest and these are the weights that regression assigned to the scores obtained basing on brief title field in VSM and LMJM. It's because this field is short and don't have as much information as the other. On the other side we can see the last two weights that correspond to the criteria field, in both models they scored high, it's because they consist only of important words, short phrases that are essential to correctly match the query. In other fields that are longer the score might be more random because of the use of stop words, irrelevant words that are frequent in English language. We can observe that looking at the 3,4 and 5,6 weights, the first pair is for brief summary and the second one is for detailed description, shorter form is favoured than more detailed one. When we take a look at the lambdas that we chose for the fields (0.10,0.25,0.25,0.15), we can see that the second and third are the highest, that means that the impact of probability of word occurrence in whole brief summary corpus and detailed description corpus is lowered. We think that it is like that to distinct the scores of the documents.

13. When do models fail

The missing fields might have a big impact on LETOR performance because we substitute them with the worse performing ones, because each document has a title, but as we can see the title field is the worst performing one, so when we calculate the score that the models obtained basing on these substituted fields it is much worse than the non substituted ones. Let's take for an example a document that has only brief title and a criteria, brief summary and detailed description are substituted with brief title, and when we calculate the scores using the logistic weights which assign high importance to these fields, poorly relevant fields (because they are based on brief title) that has a much higher impact on the final score than it should.

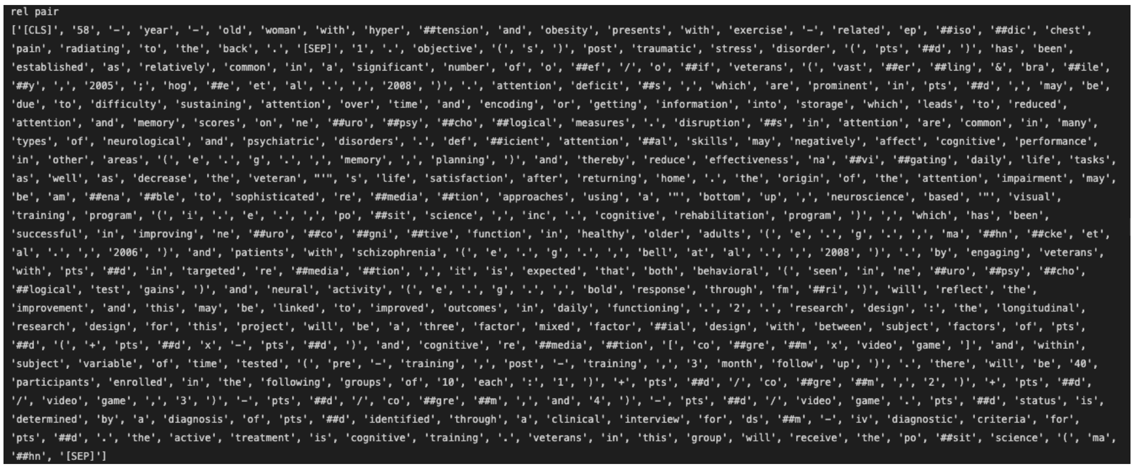
14. Phase three

In the third phase we were meant to use BERT model and extract contextual embeddings to improve the results obtained in the second phase. We have loaded the bert model into our project and started with analysis of the bert tokenizer.

15. Tokenizer

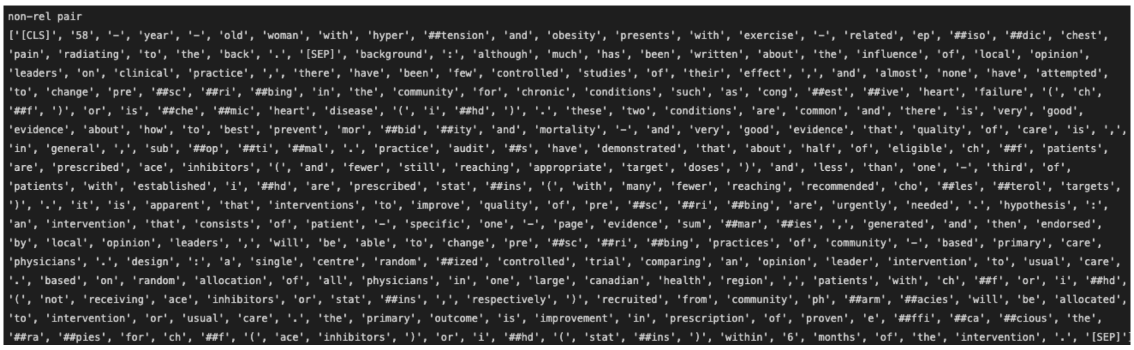
For the analysis of the tokenizer we have started with selecting pairs of queries and documents, one that was labeled as relevant pair and the other that was labeled as non relevant. We did it for 3 queries. Then we used the provided tokenizer and converted the ids that tokenizer gave as output into tokens to see in what way the input text is split.

Sample output for relevant pair:



```
rel pair
['[CLS]', '58', '-', 'year', '-', 'old', 'woman', 'with', 'hyper', '##tension', 'and', 'obesity', 'presents', 'with', 'exercise', '-', 'related', 'ep', '##iso', '##dic', 'chest',
'pain', 'radiating', 'to', 'the', 'back', '.', '[SEP]', '1', '.', 'objective', '(', 's', ')', 'post', 'traumatic', 'stress', 'disorder', '(', 'pts', '##d', ')', 'has', 'been',
'established', 'as', 'relatively', 'common', 'in', 'a', 'significant', 'number', 'of', 'o', '##ef', '/', 'o', '##if', 'veterans', '(', 'vast', '##er', '##ling', '&', 'bra', '##ile',
'##y', ',', '2005', ';', 'hog', '##e', 'et', 'al', '.', ',', '2008', ')', '.', 'attention', 'deficit', '##s', ',', 'which', 'are', 'prominent', 'in', 'pts', '##d', ',', 'may', 'be',
'due', 'to', 'difficulty', 'sustaining', 'attention', 'over', 'time', 'and', 'encoding', 'or', 'getting', 'information', 'into', 'storage', 'which', 'leads', 'to', 'reduced',
'attention', 'and', 'memory', 'scores', 'on', 'ne', '##uro', '##psy', '##cho', '##logical', 'measures', '.', 'disruption', '##s', 'in', 'attention', 'are', 'common', 'in', 'many',
'types', 'of', 'neurological', 'and', 'psychiatric', 'disorders', '.', 'def', '##icient', 'attention', '##al', 'skills', 'may', 'negatively', 'affect', 'cognitive', 'performance',
'in', 'other', 'areas', '(', 'e', '.', 'g', '.', ',', 'memory', ',', 'planning', ')', 'and', 'thereby', 'reduce', 'effectiveness', 'na', '##vi', '##gating', 'daily', 'life', 'tasks',
'as', 'well', 'as', 'decrease', 'the', 'veteran', "'", 's', 'life', 'satisfaction', 'after', 'returning', 'home', '.', 'the', 'origin', 'of', 'the', 'attention', 'impairment', 'may',
'be', 'am', '##ena', '##ble', 'to', 'sophisticated', 're', '##media', '##tion', 'approaches', 'using', 'a', '"', 'bottom', 'up', ',', 'neuroscience', 'based', '"', 'visual',
'training', 'program', '(', 'i', '.', 'e', '.', ',', 'po', '##sit', 'science', ',', 'inc', '.', 'cognitive', 'rehabilitation', 'program', ')', ',', 'which', 'has', 'been',
'successful', 'in', 'improving', 'ne', '##uro', '##co', '##gni', '##tive', 'function', 'in', 'healthy', 'older', 'adults', '(', 'e', '.', 'g', '.', ',', 'ma', '##hn', '##cke', 'et',
'al', '.', ',', '2006', ')', 'and', 'patients', 'with', 'schizophrenia', '(', 'e', '.', 'g', '.', ',', 'bell', 'at', 'al', '.', ',', '2008', ')', '.', 'by', 'engaging', 'veterans',
'with', 'pts', '##d', 'in', 'targeted', 're', '##media', '##tion', ',', 'it', 'is', 'expected', 'that', 'both', 'behavioral', '(', 'seen', 'in', 'ne', '##uro', '##psy', '##cho',
'##logical', 'test', 'gains', ')', 'and', 'neural', 'activity', '(', 'e', '.', 'g', '.', ',', 'bold', 'response', 'through', 'fm', '##ri', ')', 'will', 'reflect', 'the',
'improvement', 'and', 'this', 'may', 'be', 'linked', 'to', 'improved', 'outcomes', 'in', 'daily', 'functioning', '.', '2', '.', 'research', 'design', ':', 'the', 'longitudinal',
'research', 'design', 'for', 'this', 'project', 'will', 'be', 'a', 'three', 'factor', 'mixed', 'factor', '##ial', 'design', 'with', 'between', 'subject', 'factors', 'of', 'pts',
'##d', '(', '+', 'pts', '##d', 'x', '-', 'pts', '##d', ')', 'and', 'cognitive', 're', '##media', '##tion', '[', 'co', '##gre', '##m', 'x', 'video', 'game', ']', 'and', 'within',
'subject', 'variable', 'of', 'time', 'tested', '(', 'pre', '-', 'training', ',', 'post', '-', 'training', ',', '3', 'month', 'follow', 'up', ')', '.', 'there', 'will', 'be', '40',
'participants', 'enrolled', 'in', 'the', 'following', 'groups', 'of', '10', 'each', ':', '1', ')', '+', 'pts', '##d', '/', 'co', '##gre', '##m', ',', '2', ')', '+', 'pts', '##d',
'/', 'video', 'game', ',', '3', ')', '-', 'pts', '##d', '/', 'co', '##gre', '##m', ',', 'and', '4', ')', '-', 'pts', '##d', '/', 'video', 'game', '.', 'pts', '##d', 'status', 'is',
'determined', 'by', 'a', 'diagnosis', 'of', 'pts', '##d', 'identified', 'through', 'a', 'clinical', 'interview', 'for', 'ds', '##m', '-', 'iv', 'diagnostic', 'criteria', 'for',
'pts', '##d', '.', 'the', 'active', 'treatment', 'is', 'cognitive', 'training', '.', 'veterans', 'in', 'this', 'group', 'will', 'receive', 'the', 'po', '##sit', 'science', '(', 'ma',
'##hn', '[SEP]']
```

Sample output for non relevant pair:

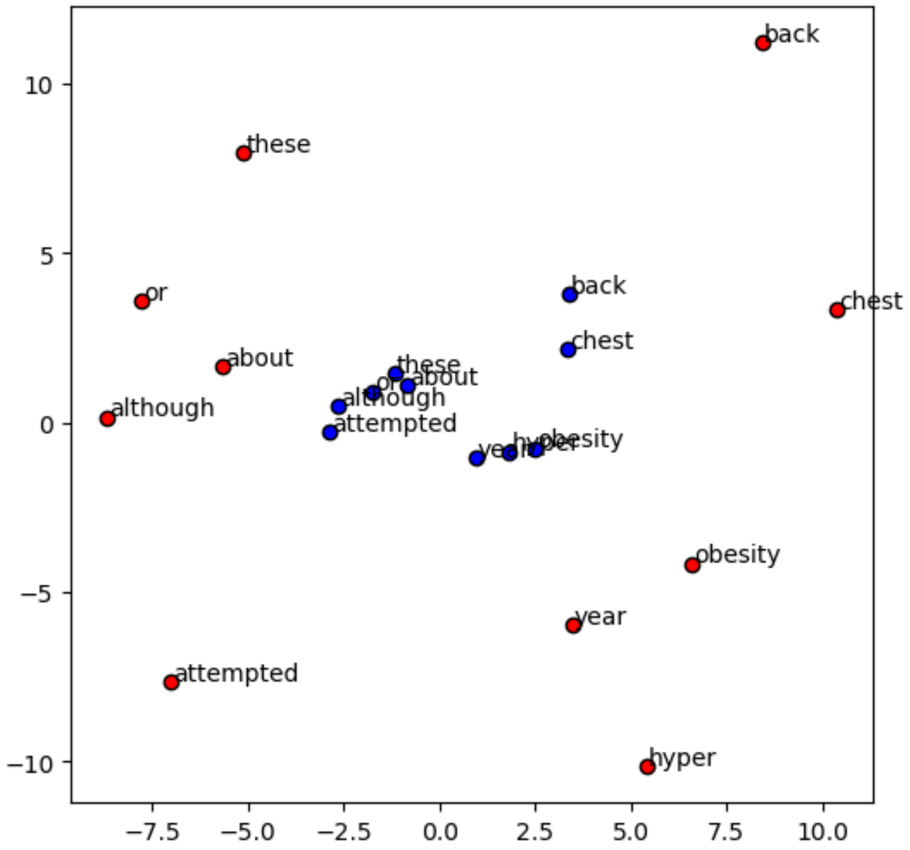


```
non-rel pair
['[CLS]', '58', '-', 'year', '-', 'old', 'woman', 'with', 'hyper', '##tension', 'and', 'obesity', 'presents', 'with', 'exercise', '-', 'related', 'ep', '##iso', '##dic', 'chest',
'pain', 'radiating', 'to', 'the', 'back', '.', '[SEP]', 'background', ':', 'although', 'much', 'has', 'been', 'written', 'about', 'the', 'influence', 'of', 'local', 'opinion',
'leaders', 'on', 'clinical', 'practice', ',', 'there', 'have', 'been', 'few', 'controlled', 'studies', 'of', 'their', 'effect', ',', 'and', 'almost', 'none', 'have', 'attempted',
'to', 'change', 'pre', '##sc', '##ri', '##bing', 'in', 'the', 'community', 'for', 'chronic', 'conditions', 'such', 'as', 'cong', '##est', '##ive', 'heart', 'failure', '(', 'ch',
'##f', ')', 'or', 'is', '##che', '##mic', 'heart', 'disease', '(', 'i', '##hd', ')', '.', 'these', 'two', 'conditions', 'are', 'common', 'and', 'there', 'is', 'very', 'good',
'evidence', 'about', 'how', 'to', 'best', 'prevent', 'mor', '##bid', '##ity', 'and', 'mortality', '-', 'and', 'very', 'good', 'evidence', 'that', 'quality', 'of', 'care', 'is', ',',
'in', 'general', ',', 'sub', '##op', '##ti', '##mal', '.', 'practice', 'audit', '##s', 'have', 'demonstrated', 'that', 'about', 'half', 'of', 'eligible', 'ch', '##f', 'patients',
'are', 'prescribed', 'ace', 'inhibitors', '(', 'and', 'fewer', 'still', 'reaching', 'appropriate', 'target', 'doses', ')', 'and', 'less', 'than', 'one', '-', 'third', 'of',
'patients', 'with', 'established', 'i', '##hd', 'are', 'prescribed', 'stat', '##ins', '(', 'with', 'many', 'fewer', 'reaching', 'recommended', 'cho', '##les', '##terol', 'targets',
')', '.', 'it', 'is', 'apparent', 'that', 'interventions', 'to', 'improve', 'quality', 'of', 'pre', '##sc', '##ri', '##bing', 'are', 'urgently', 'needed', '.', 'hypothesis', ':',
'an', 'intervention', 'that', 'consists', 'of', 'patient', '-', 'specific', 'one', '-', 'page', 'evidence', 'sum', '##mar', '##ies', ',', 'generated', 'and', 'then', 'endorsed',
'by', 'local', 'opinion', 'leaders', ',', 'will', 'be', 'able', 'to', 'change', 'pre', '##sc', '##ri', '##bing', 'practices', 'of', 'community', '-', 'based', 'primary', 'care',
'physicians', '.', 'design', ':', 'a', 'single', 'centre', 'random', '##ized', 'controlled', 'trial', 'comparing', 'an', 'opinion', 'leader', 'intervention', 'to', 'usual', 'care',
'.', 'based', 'on', 'random', 'allocation', 'of', 'all', 'physicians', 'in', 'one', 'large', 'canadian', 'health', 'region', ',', 'patients', 'with', 'ch', '##f', 'or', 'i', '##hd',
'(', 'not', 'receiving', 'ace', 'inhibitors', 'or', 'stat', '##ins', ',', 'respectively', ')', 'recruited', 'from', 'community', 'ph', '##arm', '##acies', 'will', 'be', 'allocated',
'to', 'intervention', 'or', 'usual', 'care', '.', 'the', 'primary', 'outcome', 'is', 'improvement', 'in', 'prescription', 'of', 'proven', 'e', '##ffi', '##ca', '##cious', 'the',
'##ra', '##pies', 'for', 'ch', '##f', '(', 'ace', 'inhibitors', ')', 'or', 'i', '##hd', '(', 'stat', '##ins', ')', 'within', '6', 'months', 'of', 'the', 'intervention', '.', '[SEP]'
```

As we can see the first token is CLS, this token is used for next sentence prediction and in each pair (query, document) this token is put by the tokenizer at the beginning, also one of the special tokens is SEP token that separates the query from the document and is used to mark the end of the document. Also we can observe that words that are unusual like “neurocognitive" that are not in the vocabulary of the tokenizer are split into smaller parts called morphemes and are marked with two ##, so the splitted word looks like this 'ne', '##uro', '##co', '##gni', ‘##tive’.

16. Layer embeddings

After the analysis of the tokenizer we proceeded to analysis of the embeddings of some sample tokens. We extracted the embeddings for these tokens and then compared the first and the last layer of these embeddings and plotted them using PCA to visualise how they shifted between first and last layer. Red represents the first layer

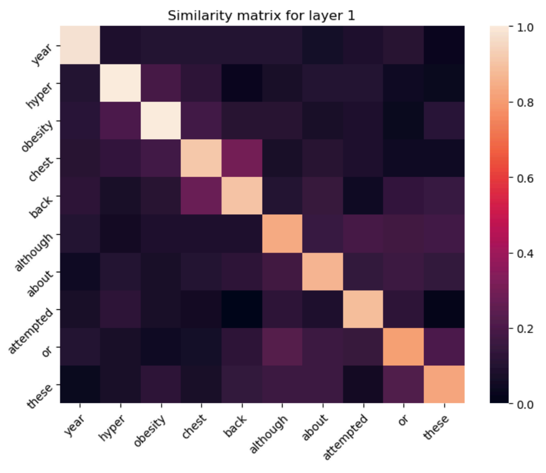


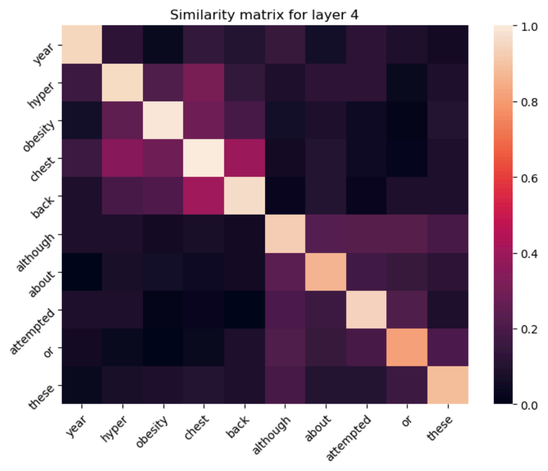
and blue the last one. We can see that embeddings shifted towards each other creating a more dense groups, what is worth noticing is that words like “although”, “about”, “these”, “or” and “attempted” formed a one group. We think that it may be this way because these words are often around the nouns and they carry similar context. Word like back and chest that are both nouns for body parts also formed a small group. We could’t find the context explanation for a group “year”, ”obesity” and “hyper” but it may be just a case that these words are all from the query and they are often used in the description of patient condition.

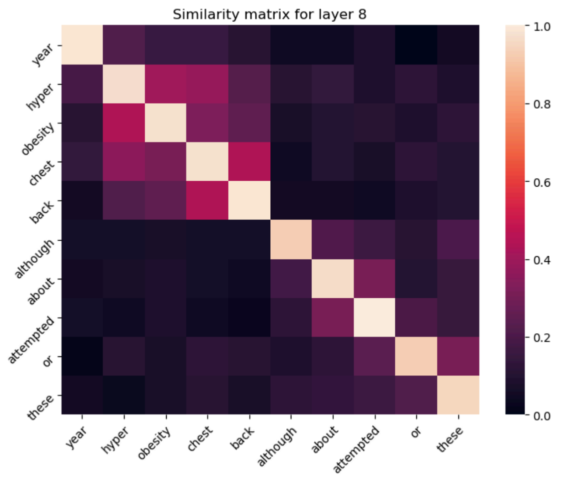
17. Flow of changes in layer embeddings

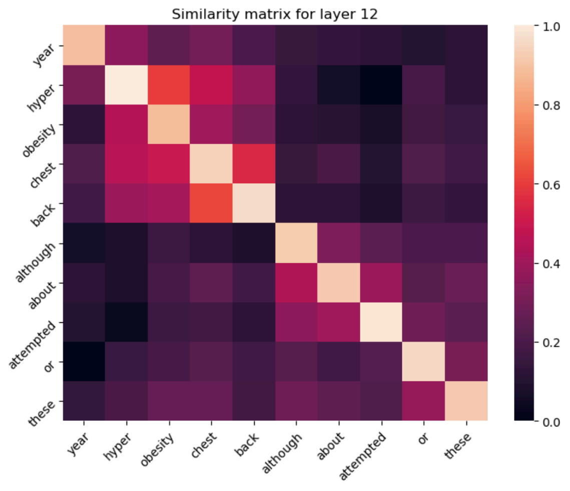
In this part we focused on the similarities between consequent embedding layers for a set of givens tokens, we wanted to observe what different layers capture and what is the flow of changes. In the obtained confusion matrices we can see the dynamics of layers and how some of the words do not change that much. We can also see how the words tend to each other forming some kind of square patterns, in the previous graph that was made using PCA we can see that these words are creating a groups, these words have similar semantic meaning or use. Also often we can observe that words that are all from the query form a group, the same things happen with document. Similarities are computed as dot products of consequent embedding layers. (The rest of visualisations is available in visualisations.ipynb)

Non relevant pair

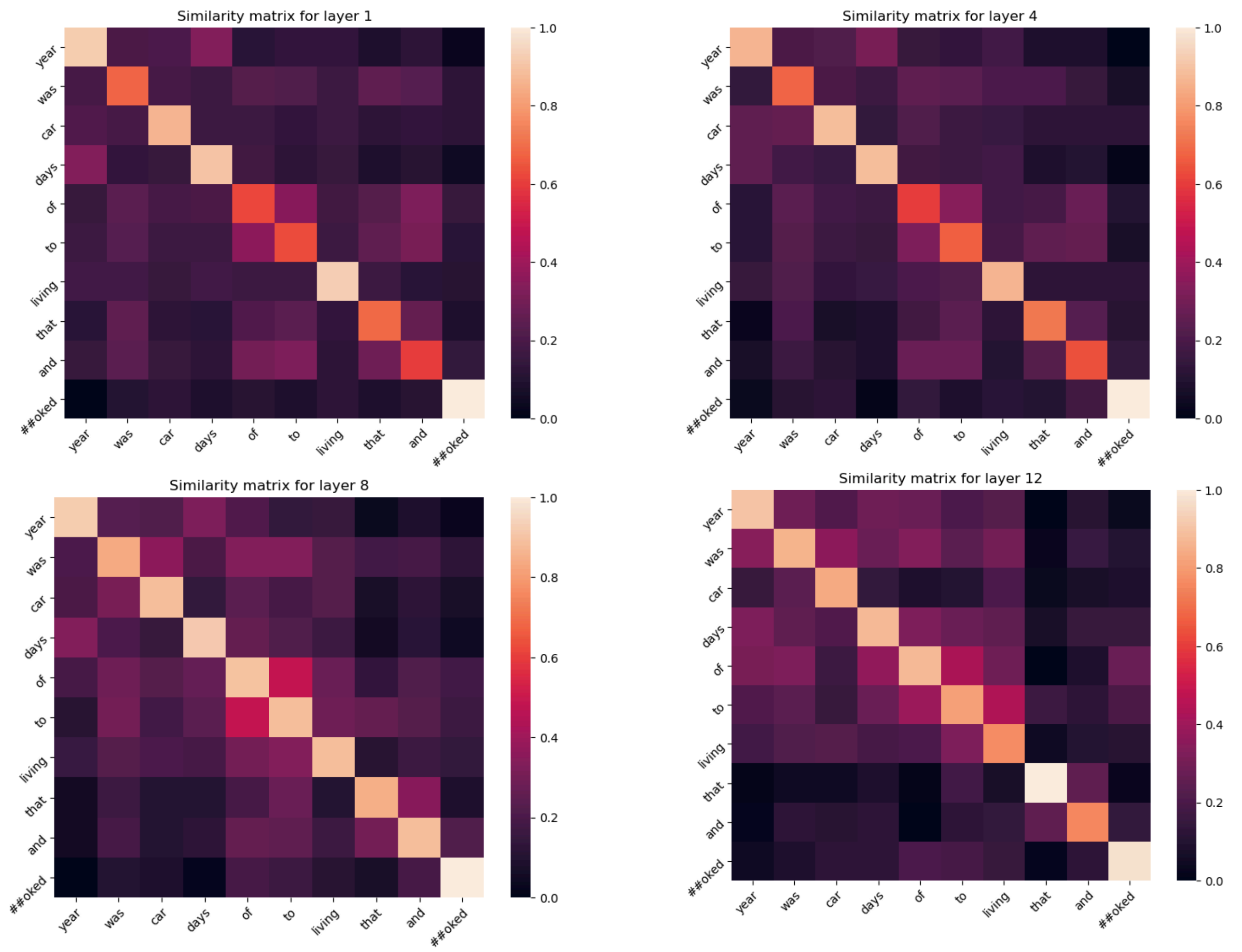






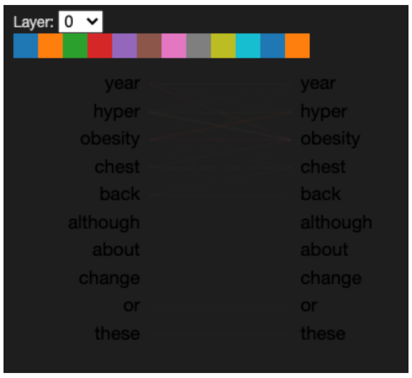


Relevant pair



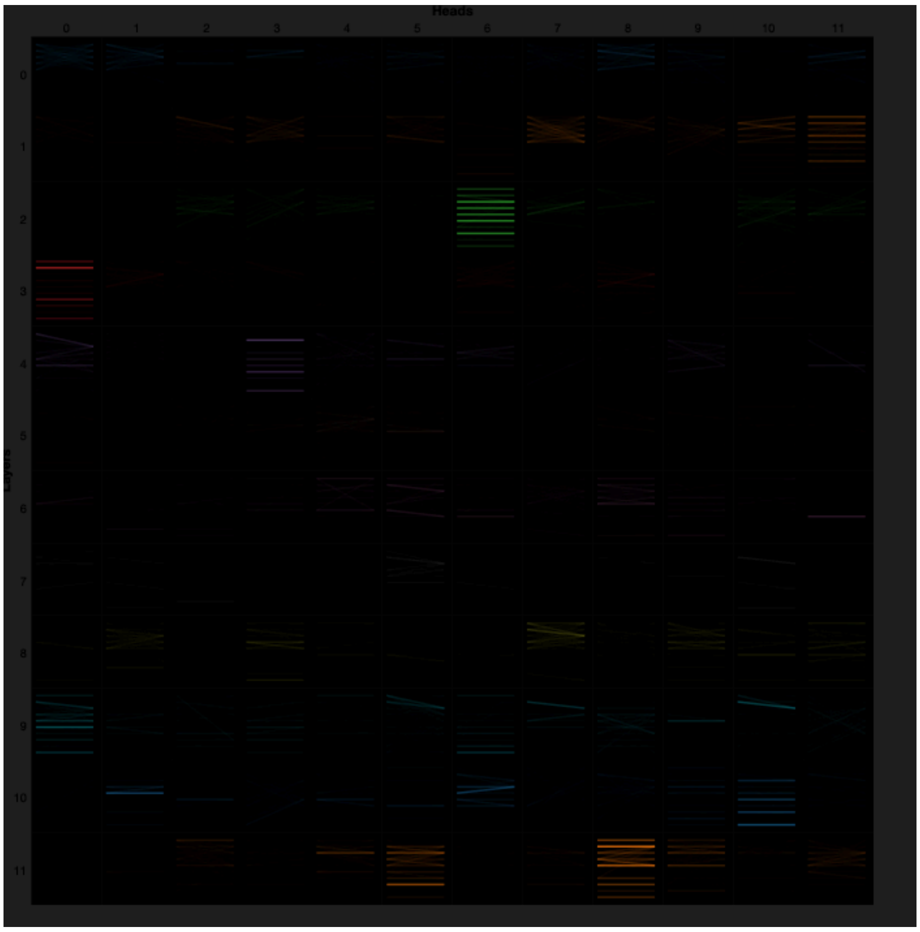
18 .Self-attention head visualization

In this visualisation we can discover which word in which layer had impact into reweighing the embeddings and how different heads focus on different tokens. We can see that the tokens that we have chosen are not influencing each other that much, and only these that are not distant in the sentences have some barely seen connection.



year year
hyper hyper
obesity obesity
chest chest
back back
although although
about about
change change
or or
these these

In the second visualisation we can see how different heads in different layers focus on different tokens. (These visualisations here are unreadable, it is possible to interact with the first one in the visualisations.ipynb)



19. Letor with embeddings

We did struggle with efficient way of extracting embeddings at first we tried to perform cross validation and for each set in k-fold we calculated CLS embeddings for pairs (query, doc) in the training set that were labelled in qrels file and then calculated the embeddings for pairs (every query, every document) in the validation set . It was horribly time consuming and inefficient because we overlapped a lot of excessive computations. When we decided to calculate embeddings for each possible combination of (query, doc) and then eventually operate on subsets of this embeddings set it was too late to make it happen.. We decided to just run one LETOR model with s regularisation parameter equal to 0.9, and compare the results with the previous phase.

We calculated the embeddings for each query in the training set and labelled documents, then we derived logistic regression coefficients and then for each query in the training set compute the CLS token embedding for all of the documents, then multiply them by previously obtained coefficients sort them and the ranking using same metrics as in the previous stages, we sum the metrics and at the end of the evaluation. After running this configuration in turned out that the results in the metrics are terrible, we were not able to find the bug despite multiple tries. The attempted LETOR with embeddings implementation is is the phase3 notebook, and all of the visualisations and tokenizer analysis in the visualisations notebook.