Reproducing experimental results from paper: Entity Set Search of Scientific Literature: An Unsupervised Ranking[1]

Reproducibility Report*

Aileen Michelle Dick
Data Science MSc. program
Vienna University of Technology
Vienna, Austria
e11706782@student.tuwien.ac.at

Lukas Prem
Data Science MSc. program
Vienna University of Technology
Vienna, Austria
e11707266@student.tuwien.ac.at

Gábor Zoltán Tóásó
Data Science MSc. program
Vienna University of Technology
Vienna, Austria
e12127079@student.tuwien.ac.at

ABSTRACT

As newcomers, our ultimate goal is to primarily investigate the reproducibility of the – above referred – scientific paper; check the consistency of the outcome; validate its correctness and accuracy; and potentially challenge the ultimate conclusions by strictly following the original execution steps. The scope of the reproducing activity is defined as part of the Exercise 2 of the lecture "Experiment Design for Data Science" in 2021WS.

The hypothesis of the original experiment was to prove that the self-developed *SetRank*-algorithm, a novel unsupervised model selection approach, significantly outperforms the current web or general domain search approaches for searching scientific literatures. Especially regarding entity-set queries modelling inter-entity relations and capturing entity type information. For the experiment, two predefined sets of data were used.

KEYWORDS

Entity-Set Aware Search; Unsupervised Ranking Model; Unsupervised Model Selection; Literature Search; Reproducibility;

ACM Reference format:

Aileen Michelle Dick, Lukas Prem and Gábor Zoltán Tóásó - 2022. *Reproducing experimental result from paper: Entity Set Search of Scientific Literature: An Unsupervised Ranking Approach*. Wien, Austria, 6 pages. DOI: 10.5281/zenodo.5920442

1 Introduction

Nowadays, it is an elementary expectation from research papers to be documented in a way that the underlying calculations and results contributing to the final conclusions are reproducible, based on documented workflows, accessible data and code basis. Here, we intend to describe the performed steps and the faced issues that we met in course of our reproducing experiment. In case of conceptual or technical gaps, we follow a conservative approach (to stay close to the anticipated, original process steps). As final step, we present the differences between the documented results and the re-calculated ones (including significance testing).

According to the original research paper, scientific literature search is different from Web or general domain search (working with short keyword queries). The majority of queries in scientific literature search are (concrete or abstract) *entity-set ones*. They reflect user's need for finding documents that (i) contain multiple entities, (ii) reveal inter-entity relationships and (iii) the query provides a ranked list of documents that are most relevant to the whole entity set – in contrast to supervised approaches that model each entity separately (based on click history). *SetRank*, unsupervised ranking framework is developed to model inter-entity relationships and captures entity type information based on the technique of weighted rank aggregation. It first links entity mentions in query and documents to an external knowledgebase. Then, each document is represented with both *bag-of-words* and *bag-of-entities representations* and fits two language models respectively. A novel heterogeneous graph representation models complex entity information (e.g., entity type) and entity relations within the set. Then, the query-document matching is defined as a graph covering process and each document is ranked based on the information need it covers in the query graph. Some parameters still need to be learned using a labelled validation set.

The main conclusions are made, by reporting the implemented SetRank algorithm to four different, commonly used baseline models:

- BM25: Vector space model
- LM-DIR: Query likelihood model using Dirichlet Smoothing
- LM-JM: Query likelihood model using Jelinek Mercer Smoothing
- IB: Information-based model

Calculations are done for a biomedical genomics track (TREC-BIO) dataset, as well as for Semantic Scholar query data (S2-C2) covering the field of Computer Science.

The algorithms are then compared with the Normalized Discounted Cumulative Gain (NDCG), which is an adequate measure for the performance of recommender systems.

The remaining part of this report is organized as follows: In Section 2 related strategic approach for the reproducibility are discussed. Section 3 presents the faced difficulties over the experiments in connection to the execution process steps. Section 4 shows the comparison of the reproduced experimental results on two benchmark datasets. Finally, Section 5 concludes this work with discussions about the results.

2 Reproducibility approach

In course of specifying the experiment design, we tried to stay as close as possible to the original implementation settings.

First, we needed to get familiar with the key aspects of the hypothesis setting in the original concept paper and the experiment workflow described in the *README* files stored in the author's GitHub repository. The original code repository has been cloned to our own repository in case for any code adjustment[1].

2.1 Technical framework

Before starting to reproduce results, an Elasticsearch engine needs to be setup. Elasticsearch is a widely used search and analytics engine, which comes especially powerful with huge amounts of data. It can be downloaded from the official elasticsearch website[2], fortunately the version 5.4.0, used for the workflow, is still available. In addition, a few packages should be installed that are stated in the requirements for running the baseline and SetRank scripts:

- Elasticsearch 5.4.0
- Python 3.7
    - Elasticsearch 5.4.0
    - Textblob 0.13.0

After downloading elasticsearch, the following lines need to be inserted into the underlying yaml-file in ***config/elastisearch.yml***, in order to function properly:

- script.inline: true
- script.indexed: true

To start the engine, the batch-file ***bin/elasticsearch.bat*** needs to be executed afterwards in a terminal window.

Additionally, a tool called "pytrec_eval" is used for evaluation. Fortunately, the version was cloned into their GitHub repository. There were no set-up issues here, other difficulties using *pytrec_eval* are described in section 3.

2.2 Input data

2.2.1 Source data

The original research paper evaluated the proposed unsupervised approach using TREC Genomics Tracks (abbreviated TREC-BIO) data and Semantic Scholar's query log (abbreviated S2-CS) data. This data was not included in this repository. However, the latest weblinks were provided in the README, such that the TREC-BIO data could be downloaded from a google drive folder[3], whereas the S2-C2 was found on a website[4].

2.2.1 Query data

The calculations will be done based on the input queries for both data sets, */data/S2-C2/s2_query.json* for the S2-C2 data and */data/TREC-BIO/trec_query.json.* In the paper it was mentioned, that 40 out of all 100 queries are entity-set queries for S2-C2. Therefore, this json file needed to be manually split into a file for word queries and one for entity-set queries. This could be achieved by looking at the "ana" key in each line. Ff there was more than one linked entity reference in there, we consider it as entity-set.

1) {"qid": "21", "ana": {"/m/03j0x": 1}, "query": "human computer interaction"}
2) {"qid": "22", "ana": {"/m/064dh3": 1, "/m/031f5p": 1}, "query": "eye movement clustering"}

These are two of the queries from the json file as an example. Number 1 would be considered as word representation, whereas number 2 would be considered as entity representation.

The same strategy is applied to the TREC-BIO data. There 86 out of 100 queries can be observed as entity-set queries.

2.3 Code

In the mentioned GitHub repository, python scripts for all baseline models and the SetRank algorithm can be found. In addition, the calculations for S2-C2 and TREC-BIO are located in separate folders.

2.3.1 Prerequisites

Some adaptions need to be done beforehand in the code. There have been inconsistencies in the given paths. The file actually called TREC-BIO in the data and baseline folder was incorrectly referenced as TREC_BIO.

Moreover, to be able to run *setrank_TREC.py*,

- import nltk
- nltk.download('punkt')

needs to be imported additionally in this file.

[1] https://github.com/gabortoaso/G10-Research-paper-validation
[2] Download Elasticsearch | Elastic
[3] SetRank-dataset – Google Drive
[4] Explicit Semantic Ranking Dataset — Allen Institute for AI (allenai.org)

2.3.2 Index

Before running the algorithm, the appropriate indexes need to be created on the engine. For that purpose, the ***create_index.py*** should be executed for all four baseline models, BM25, LM-DIR,LM-JM,IB separately, as well as for the SetRank algorithm itself, and everything for both datasets. The index creation files for the baselines are stored in *code/baselines/S2-C2,* and *code/baselines/TREC-BIO.* The index will be created for a specified model by adding the argument -sim to the execution, like: ***python create_index.py -sim "bm25"***. For SetRank calculations, the index creation file stored in *code/SetRank* and can be called without any arguments: ***python create_index.py.***

These indexes are just like a skeleton and need to be filled with the data in a next step. For that purpose, the ***index_data.py*** should be run. Again, for all algorithms and datasets respectively. The scripts are located in the exact same folders specifies previously. For the BM25 baseline the command would be: ***python index_data.py -sim "bm25"***. For the SetRank indexes, again no sim-argument has to be specified.

Just after this phase, the actual calculations on the data can be done.

2.3.3 Main calculations

The results for the baseline models using the search engine are produced by ***code/baselines/S2-C2/search_data.py*** and ***code/baselines/TREC-BIO/search_data.py.*** These scripts take two additional arguments, the model specification again and the mode of querying, which is either "word" for bag-of-words models, "entity" for bag-of-entities, or "both", the combination the two approaches. For BM25 for example, the following command should be executed:

- python search_data.py -sim "BM25" -mode "word"
- python search_data.py -sim "BM25" -mode "entity"
- python search_data.py -sim "BM25" -mode "both"

These steps need to be performed for all 4 different baselines and both datasets.

The SetRank code is located in different files again. For S2-C2, ***code/SetRank/setRank_ESR.py*** should be considered, and for TREC data ***code/SetRank/setRank_TREC.py*** **.** There would be an option for specifying arguments again, the input, output file and parameters, but by default they are already provided. The restructured queries are the input for the calculations, and the results are saved as .run files at the specified location.

2.3.3.1 Parameter Tuning

Separate files for tuning modelling parameters for the SetRank algorithm are given. It can be computed by either executing ***code/SetRank/autoSetRank_ESR.py*** for the S2C2 data, or ***code/SetRank/autoSetRank_TREC.py*** for TREC-BIO data, where 5-fold cross-validation using the NDCG@20 score has been implemented.

2.3.4 Generating and evaluating output results

For calculating the measures of success, a shell script called *eval.sh* was provided in the folder *pytrec_eval/examples*. In this file we had to modify the paths. It is a simple script which calls *trec_eval.py*, writes the results to a file and then parses them to create a more readable output which is again written to a file. Calling it for our SetRank results for example looks like this:

```
./eval.sh ../../results/S2C2/setRank_both.run
setRank_both
```

The corresponding results are presented in Table 1.

After evaluation, adequate statistical tests have been conducted.
For statistical significance tests there was also a python script provided in *pytrec_eval/examples*. This script had to be called supplying the results of two different algorithms (e.g. setRank against the results of one of the baselines) and the corresponding *.qrel* file, depending on the dataset used. Also, with *--measure* a measure had to be defined and even with a list of options printed out if you called the file without supplying a measure, it took as trial and error to find out that you could append *ndcg_cut* with *_5, _10, _15* or *_20* to get the p-value for each specific measure.

Since we wanted to test all of our results against all corresponding baselines (e.g. setRank_both against bm25_both, setRank_entity against bm25_entity, etc.), we wrote two short scripts which iterate over all four baseline algorithms, where we call for each of *_both, _entity* and *_word* another shell script which we wrote which internally calls the *statistical_significance.py* script with the aforementioned four different NDCG measures. This gave us the results displayed in Table 2.

3 Difficulties

The first unexpected exercise has been to find out that an elasticsearch engine was the basis of this whole process, which has not been clear to us based on the paper and available resources. In addition, as no one of us had experiences with the engine, we first needed to read up on it and go through a cumbersome installation defined by many trials and errors, since it did not simply work after installation. As described in section 2.1, adaptions in the configuration of the elasticsearch were necessary.

Once elasticsearch was setup, some inconsistencies in the code have been detected, like wrong file paths as described in section 2.3.1.

After collecting all the python-based code files, getting the overall structure was challenging as well, since the few independent README's did not give a clear overview of what is what exactly, how files should be executed and in which order.

Since the original data has not been included in the author's repository, we expected that it will not be necessary for getting the

key results. Nevertheless, it is needed to create the indexes properly in the elasticsearch engine, thus it had to be downloaded in addition by us.

Furthermore, it was obvious how to get to the separate word and entity models for the baseline algorithms. However, the process for SetRank was not conclusive. After investigation and logical connections, we came to the assumption that the queries for the SetRank algorithm where together in one file and needed to be split manually into separate files in order to execute separate word and entity models.

For evaluating the results another project named *pytrec_eval*[5] was included in the repository. This project is basically a python API to a tool called *trec_eval*, which itself is described by its authors as "[…] the standard tool used by the TREC community for evaluating an ad hoc retrieval run, given the results file and a standard set of judged results."[6] The authors of the Paper we aimed to reproduce included all the content of *pytrec_eval* repository and with that also the content of the *trec_eval* repository in their GitHub repository. This led to a bit of confusion regarding the process of setting up the evaluation tool because the README files for each of the used projects were also included and it was not clear which of the steps described in them were actually necessary to complete.

To make sure everything runs smoothly, and we do not overlook any steps we tried to do all the steps from the very beginning, which included compiling the C files of the *trec_eval*. This was the first problem we ran into, as the C Files did not compile out of the box and threw many errors. At that point we decided to just go with the already compiled executables included and try one level higher by running *setup.py* according to the README file of *pytrec_eval*. This also did not run, and we speculated that it was due to it requiring old versions of python packages, which we unfortunately were unable to install.

We ended up just being able to use *pytrec_eval*, and through that *trec_eval*, with the executables and the setup provided. This is a big problem for reproducibility because, as we will discuss in the following paragraphs, all of measures of success as well as the statistical tests are calcuated using *pytrec_eval* and if we cannot compile it ourselves, we really have no possibility to check what it is actually doing. For all we know the executable could be something that has nothing to do with the C code included in their repository. To be fair it would probably be possible to setup a system with all the old versions of python packages such that *pytrec_eval/setup.py* could run but getting compiler errors from the C files with the given makefile means that some major changes to the code would be necessary to compile it and thus make sure we know exactly what is going on in *trec_eval*.

[5] https://github.com/cvangysel/pytrec_eval
[6] https://github.com/usnistgov/trec_eval

We were ambitious to execute more profound validation of the concept, i.e. parameter tuning, testing different significant level, or constructing queries on our own. However, based on the limited timeframe and the show-stopper technical obstacles at the beginning of our reproducibility journey, we needed to **moderate our initial goal**. Alternatively, we focused on **investigating the reproducibility of the outcome of the published scientific paper** (i.e. to confirm the numbers and findings reported in the paper).

The authors have spent some effort to document the key steps of the experiment design, however minor but still important steps have been kept on significantly different granularity level. It might come from some generic assumptions from the author perspective, which resulted in a non-comprehensive list of instructions. Basically, our initial conclusion is that the quality of the **experiment process-flow was not adequately documented,** neither in the research paper nor in the git-hub repository. This fact required us to spent significant amount of time until we figured out the missing points.

We also considered reaching out to the authors of the original concept paper, but based on the git-hub issue status, they do not seem to be reachable anymore concerning this project. So, we needed to accept limited possibility for additional or external consultancy.

4 Results and Key findings

Here, we would like to show the outcome of the comparison of the original and the reproduced output results. Additionally, we intend to materialize whether the differences are significant (questioning the original conclusions of the hypothesis) based on statistical significance testing.

4.1 Evaluation on metrics

As an outcome of the previously mentioned steps, we have calculated NDCG, MAP and success measures for all approaches. Metrics have been computed considering the 5, 10, 15, and 20 highest rankings respectively

As in the paper, we chose to compare NDCG values for final conclusions. In Table 1, original obtained results are shown at the top, our reproduced results are shown in the middle, and at the bottom the differences for each cell are shown. For quick comparisons, all scores that differ within four decimal places are color-coded - negative scores are highlighted green, meaning the reproduced results perform better, whereas positive scores are highlighted in orange, i.e. the originally obtained results perform better.

The overall picture is that for all baseline models, we were able to get almost the same results. Only a few seem to be differentiating. Interestingly, the majority of differences are cause by the NDCG@20 scores, where NDCG 5 to 15 scores are not differing. For the SetRank algorithm, results for all three representations, word, entity, or both, differ. Moreover, it is visible that for entity-

set representations the score has improved. On the other hand side, the scores of the combination of both methods have been decreasing.

4.2 Significance testing

Pairwise t-tests have been computed using the pytrec_eval tool as well, to get an understanding of significant coherences between the SetRank algorithm and other baseline models. Findings are reported in Table 2, where all p-values smaller than 0.05 have been coloured.

For the S2C2 dataset the SetRank algorithm clearly performs significantly better than all other baseline models. Just for some models based on NCDG5 it did not improve, and for the IB algorithm using both representation it just performed significantly higher with NDCG@20 score and not with the others.

Considering TREC-BIO data, p-values are truly small for all entity-set models, which means that there is a significant improvement using the implemented SetRank approach. The opposite case is for all word representations, where the p-values are constantly high. There and for the IB model case both, the SetRank algorithm did not perform significantly better.

According to our NDCG scores, the SetRank algorithm outperforms all other baseline models for both datasets, which exactly matches the inference of the original paper as well.

5 Conclusions

After reviewing the concept, trying several settings for the technical setup and evaluating the final results of "Entity Set Search of Scientific Literature: An Unsupervised Ranking Approach", we came to the following conclusions.
The concept of SetRank was adequately documented in the report; however, the associated source code (in GitHub repository) and calculation workflows were not executable straight away. Some adaptions were necessary to be done, that we needed to figure out by ourselves after some time-consuming iterations. Based on the final results, we concluded that the algorithms delivered very similar results to the originally reported ones. However, some steps were not 100% comprehensible from the available descriptions, which could have led to the slightly diverging results. Furthermore, the parameter tuning was questionable, where with some improvements or adaptions better results could be potentially produced.
We made our code, workflow description and recalculation result available in our GitHub repository[1].

REFERENCES

[1] Jiaming Shen, Jinfeng Xiao, Xinwei He, Jingbo Shang, Saurabh Sinha, Jiawei Han. 2018. *Entity Set Search of Scientific Literature: An Unsupervised Ranking Approach*. In SIGIR '18: The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, July 8– 12, 2018, Ann Arbor, MI, USA. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209978.3210055

ORIGINAL		BM25			LM-DIR			LM-JM			IB			SetRank		
Dataset	Method	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both
S2-CS	NDCG@5	0,3476	0,3319	0,3675	0,3447	0,3460	0,3563	0,3626	0,3394	0,3625	0,3759	0,3420	0,3729	0,3890	0,3761	0,4207
	NDCG@10	0,3785	0,3520	0,4039	0,3623	0,3579	0,3901	0,3774	0,3519	0,3962	0,3903	0,3557	0,4009	0,4168	0,3885	0,4431
	NDCG@15	0,4001	0,3616	0,4160	0,3781	0,3673	0,4077	0,4051	0,3666	0,4174	0,4113	0,3699	0,4272	0,4411	0,4054	0,4762
	NDCG@20	0,4126	0,3752	0,4333	0,4012	0,3816	0,4205	0,4182	0,3804	0,4362	0,4295	0,3855	0,4421	0,4674	0,4229	0,4950
TREC-BIO	NDCG@5	0,3189	0,1542	0,2613	0,3053	0,1755	0,2669	0,2957	0,1656	0,2826	0,3045	0,1842	0,2770	0,3417	0,2111	0,3744
	NDCG@10	0,2968	0,1488	0,2472	0,2958	0,1601	0,2571	0,2742	0,1588	0,2572	0,2918	0,1715	0,2633	0,3165	0,1976	0,3522
	NDCG@15	0,2833	0,1424	0,2395	0,2852	0,1579	0,2591	0,2642	0,1575	0,2437	0,2835	0,1664	0,2541	0,3017	0,1931	0,3363
	NDCG@20	0,2739	0,1419	0,2337	0,2781	0,1558	0,2547	0,2560	0,1534	0,2362	0,2722	0,1628	0,2406	0,2900	0,1885	0,3246
RE-CALCULATED		**BM25**			**LM-DIR**			**LM-JM**			**IB**			**SetRank**		
Dataset	Method	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both
S2-CS	NDCG@5	0,3476	0,3319	0,3675	0,3447	0,3460	0,3563	0,3626	0,3394	0,3625	0,3759	0,3420	0,3729	0,3674	0,4232	0,3897
	NDCG@10	0,3785	0,3520	0,4039	0,3623	0,3579	0,3901	0,3774	0,3519	0,3962	0,3903	0,3557	0,4009	0,3928	0,4471	0,4145
	NDCG@15	0,4001	0,3619	0,4160	0,3781	0,3673	0,4077	0,4051	0,3666	0,4174	0,4113	0,3699	0,4272	0,4266	0,4657	0,4422
	NDCG@20	0,4126	0,3754	0,4331	0,4008	0,3812	0,4203	0,4182	0,3794	0,4352	0,4292	0,3853	0,4413	0,4420	0,4858	0,4595
TREC-BIO	NDCG@5	0,3189	0,1542	0,2613	0,3053	0,1755	0,2669	0,2957	0,1656	0,2826	0,3045	0,1842	0,2770	0,3024	0,3105	0,3093
	NDCG@10	0,2968	0,1488	0,2472	0,2958	0,1601	0,2571	0,2742	0,1594	0,2572	0,2918	0,1715	0,2633	0,2960	0,2857	0,2872
	NDCG@15	0,2833	0,1424	0,2395	0,2852	0,1579	0,2591	0,2642	0,1580	0,2437	0,2835	0,1664	0,2541	0,3069	0,2738	0,2784
	NDCG@20	0,2739	0,1421	0,2337	0,2781	0,1558	0,2547	0,2560	0,1539	0,2362	0,2722	0,1631	0,2406	0,2980	0,2641	0,2689
DIFFERENCES		**BM25**			**LM-DIR**			**LM-JM**			**IB**			**SetRank**		
Dataset	Method	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both
S2-CS	NDCG@5	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0216	-0,0471	0,0310
	NDCG@10	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0240	-0,0586	0,0286
	NDCG@15	0,0000	-0,0003	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0145	-0,0603	0,0340
	NDCG@20	0,0000	-0,0002	0,0002	0,0004	0,0004	0,0002	0,0000	0,0010	0,0010	0,0003	0,0002	0,0008	0,0254	-0,0629	0,0355
TREC-BIO	NDCG@5	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0393	-0,0994	0,0651
	NDCG@10	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	-0,0006	0,0000	0,0000	0,0000	0,0000	0,0205	-0,0881	0,0650
	NDCG@15	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	-0,0005	0,0000	0,0000	0,0000	0,0000	-0,0052	-0,0807	0,0579
	NDCG@20	0,0000	-0,0002	0,0000	0,0000	0,0000	0,0000	0,0000	-0,0005	0,0000	0,0000	-0,0003	0,0000	-0,0080	-0,0756	0,0557

Table 1: Comparison of recalculated results of SetRank with the reported performance of SetRank (orange: >0, green: <0)

RE-CALCULATED		BM25 (SetRank)			LM-DIR (SetRank)			LM-JM (SetRank)			IB (SetRank)		
Dataset	Method	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both	Word	Entity	Both
S2-CS	NDCG@5	0,0316	0,0343	0,1372	0,2504	0,0232	0,0079	0,1188	0,0782	0,0790	0,7852	0,1115	0,2595
	NDCG@10	0,0391	0,0245	0,8719	0,0102	0,0057	0,0078	0,0284	0,0152	0,1182	0,2793	0,0172	0,2230
	NDCG@15	0,0012	0,0100	0,0168	0,0002	0,0011	0,0000	0,0038	0,0121	0,0258	0,0203	0,0097	0,1044
	NDCG@20	0,0006	0,0020	0,0105	0,0010	0,0004	0,0000	0,0010	0,0029	0,0150	0,0195	0,0026	0,0319
TREC-BIO	NDCG@5	0,9341	0,0000	0,0277	0,3248	0,0000	0,0455	0,6529	0,0000	0,2003	0,9703	0,0000	0,0986
	NDCG@10	0,5259	0,0000	0,0240	0,2111	0,0000	0,1113	0,2420	0,0000	0,0911	0,7320	0,0000	0,1511
	NDCG@15	0,3697	0,0000	0,0201	0,2186	0,0000	0,2382	0,1521	0,0000	0,0353	0,2137	0,0000	0,1236
	NDCG@20	0,3016	0,0000	0,0263	0,1862	0,0000	0,3834	0,1351	0,0000	0,0399	0,2447	0,0000	0,0684

Table 2: p-values of the statistical significance testing (red: >0,05)