TREC 2017 Precision Medicine - Medical University of Graz

Pablo López-García*, Michel Oleynik, Zdenko Kasáč, Stefan Schulz

Institute for Medical Informatics, Statistics and Documentation
Medical University of Graz
Auenbruggerplatz 2 - 8036 Graz (Austria)

Corresponding author: pablo.lopez@medunigraz.at

ABSTRACT

In this paper we report on our participation in the TREC 2017 Precision Medicine track (team name: *imi_mug*). We submitted 5 fully automatic runs to both the *biomedical articles* and *clinical trials* subtasks, focusing strongly on the former. Our system was based on Elasticsearch, whose queries were generated modularly via our own open source framework. Our results showed that a modern search engine with an advanced query language is a powerful solution for the proposed tasks but it requires deep medical knowledge and careful tuning to get top performance.

INTRODUCTION

The goal of the TREC 2017 Precision Medicine track was to improve search for clinicians treating cancer patients. A set of cases describing potential patients, termed *topics*, were provided as input and contained four dimensions or features: *disease, gene, demographic,* and *other*. The challenge was divided into two subtasks that consisted of (1) retrieving relevant *biomedical articles* for treatment (from PubMed as well as ASCO and AACR conference proceedings) and (2) retrieving relevant *clinical trials* for enrollment (from ClinicalTrials.gov). A detailed description of the challenge, the datasets, and the relevance judgment guidelines are available online[1].

Experience in previous projects suggested that a search engine with an advanced query language could be an effective solution for the proposed tasks. Following this approach, once the data have been conveniently indexed in the search engine, the problem to be solved is how to transform each input topic into a query that retrieves relevant documents.

SYSTEM OVERVIEW

As the cornerstone of our system we selected Elasticsearch[2] 5.4.0, an open source search engine with a rich and fine-grained query language[3]. The preprocessing step consisted of indexing all target collections (around 27M documents), which took around 3 hours using a standard laptop. To facilitate query building in later steps, we parsed the documents to extract as much structured information as possible. This information was stored as separate fields in the index, for example as a list of MeSH tags for the PubMed abstracts, or minimum and maximum age for the clinical trials. **Appendix A** contains a summary of our final indices and a sample document.

[1] http://www.trec-cds.org/2017.html

[2] https://www.elastic.co/products/elasticsearch

[3] https://www.elastic.co/guide/en/elasticsearch/reference/current/_introducing_the_query_language.html

To generate queries automatically, we built a framework that could easily be adapted and reused for similar TREC tasks. The framework is publicly available under the terms of the MIT license[4]. Using the framework, valid Elasticsearch queries were created by choosing a *query template* and applying *query decorators* to it. *Query templates* defined search strategies and tuning parameters that proved valuable (e.g., *"the topic's disease must appear in the document's title"*). *Query decorators* allowed for the modification of topic dimensions (e.g., *"replace the topic's disease with all its known synonyms"*). Further information on the framework, query templates, and decorators is available in **Appendix B**.

STRATEGIES AND RESOURCES

This section lists the most relevant strategies and approaches that we tried using the query templates and decorators introduced above.

Query Structure

Must/should. We explored `must` and `should` clauses when running boolean queries in Elasticsearch. The former provided better results but restricted their total number, suggesting we also tried relaxing criteria (for example, moving the gene to a `should` clause).

Document fields. Virtually every template considered matching *disease* and *gene* in the *title*, *abstract*, *keyword* and *MeSH* tags in the biomedical articles subtask. For clinical trials, we took advantage of the structured fields *age* and *sex* via Elasticsearch's `range` clause, and searched for *disease* and *gene* in *title*, *summary*, and *inclusion criteria*. In most cases we down-weighted documents where any of the comorbidities from the *other* field matched the extracted exclusion criteria.

Document Type. For the biomedical articles subtask, we hypothesized that proceedings from the ASCO and AACR conferences (focused on cancer) would often be more relevant than generic PubMed articles and therefore tried boosting them.

Keywords and Boosting

Topic-oriented. As the task was focused on cancer and genetic variants, we manually added and boosted keywords and synonyms relevant to the task: *cancer*, *carcinoma*, *tumor, gene*, *genotype*, *DNA*, and *base*.

TREC extra topics. TREC provided a small reference standard with extra topics and associated relevant PubMed IDs[5]. We assumed that frequent keywords in those abstracts would also be relevant. Therefore, we retrieved the abstracts, sorted terms by frequency, and collected keywords that were general and useful for the task (e.g. *survival*, *patients*, *study*).

Medical knowledge. We also included terms suggested by a medical expert in our team or that we identified during manual inspection of the results for the reference standard creation.

Chemotherapy suffixes. We explored collecting common chemotherapy drug suffixes to boost results focused on treatment. We compiled a list of chemotherapy drug names from the code L section (L01 to L04) of ATC[6], which

[4]https://github.com/bst-mug/trec2017/

[5]http://www.trec-cds.org/extra_topics2017.pdf
[6]https://en.wikipedia.org/wiki/ATC_code_L

contains common antineoplastic and immunomodulating agents, and 200 experimental cancer drugs[7]. Then, we counted the frequency of all 3-character suffixes and added the 11 most common suffixes to the system. We later excluded the ones that increased the number of false positives (*-ide, -ine, -tat, -tin*). The final list of candidates included the suffixes *-ate, -cin, -lin, -mab, -mus, -nib,* and *-one*.

Query Expansion

Disease Expansion. As a disease can be referred to in many ways, we tried expanding the topic's disease dimension. We consulted an API-based proprietary knowledge graph of medicine[8] largely based on SNOMED CT, MeSH, and ICD, and retrieved all synonyms for the best matching concept. For example, *Cholangiocarcinoma* was expanded to additionally include *cholangiocellular carcinoma, cholangiocarcinoma of biliary tract, bile duct carcinoma,* and *bile duct adenocarcinoma.*

Gene Expansion. Following the same reasoning described above, we consulted the *Homo Sapiens* gene list provided by the National Center for Biotechnology Information[9] to expand gene names.

INTERNAL EVALUATION

As the tasks were completely unsupervised, for the biomedical articles subtask we created an internal reference standard for its evaluation, consisting of 739 documents. These documents included the top ten most relevant documents for each topic using our best strategy, as well as additional relevant documents found using PubMed, Google, and side experiments. The reference standard was built and modified iteratively every week until our runs were submitted. The relevance of 46 documents was judged by at least two annotators, and 20 were judged by three annotators. Annotators included two medical doctors and one computer scientist. The reference standard is also available under the MIT license[10].

RUNS AND RESULTS

We submitted 5 fully automatic runs to each subtask. At the conference (one topic judgment was missing), we ranked among the top 3 participants in the *biomedical articles* subtask for all metrics (infNDCG: 0.4088, R-Prec: 0.2743, P@10: 0.6172) and below 10th in the *clinical trials* subtask. The judgment order in which we submitted our runs correlated highly with the performance of the runs in the final results, indicating that our intuitions were correct. Further details on runs, results, and the strategies that were implemented can be found in **Appendix C**.

DISCUSSION

Through running our experiments we found some important limitations. Firstly, we could not always find the maximum number of results allowed for submitted runs (1,000 documents per topic), possibly because our criteria were too strict. However, a comprehensive search engine would ideally retrieve all allowed results, even if some of them were only partially relevant. We tried to overcome this limitation experimenting with a huge synonym, hypo- and hypernym list built out of the MeSH structure with the

[7]https://en.wikipedia.org/wiki/Category:Experimental_cancer_drugs
[8]https://docs.lexigram.io
[9]ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/GENE_INFO/Mammalia/Homo_sapiens.gene_info.gz
[10]https://github.com/bst-mug/trec2017/blob/master/src/main/resources/gold-standard/

mesh2solrsyn tool[11]. However, this approach pulled documents describing rare and irrelevant diseases to the top as a side effect, probably due to the idf ranking, so we eventually discarded it. Secondly, we assumed that using stemming and a stopword list would improve our metrics and tested Elasticsearch's `english` analyzer for that task. However, this was not the case, so this strategy was also discarded. The use of tiered indices and pseudo-relevance feedback are possible directions that could be investigated.

Another strategy we thought might be useful was to fine tune the Okapi BM25 ranking algorithm used by Elasticsearch with a subset of the final reference standard, in a classical supervised approach. We noticed that documents with empty abstracts tended to rank higher due to the field-length norm, even though these documents were typically irrelevant. Therefore, we explored using a modification of BM25 called BM15, which sets the field-length normalization parameter (B) to 0. Unfortunately, that worsened our overall results and led to many long abstracts being ranked higher, which was not desirable either. As we did not have a comprehensive reference standard to test different parameters, we also discarded this strategy.

We also realized that strategies were spread among experiments, templates, and decorators, which often made those strategies difficult to track and reproduce. Therefore, as future work we plan to make our framework command-line callable and more orthogonal, so that each strategy is mapped to a single modular unit.

CONCLUSION

In this notebook we reported on our participation in the TREC 2017 Precision Medicine track and described our approach, strategies, results, and lessons learnt. Our solution was based on Elasticsearch and automatic query generation using modular components, which we called query templates and query decorators. Experiments were evaluated iteratively and improved using a reference standard built specifically for the precision medicine track.

Our results confirmed that a search engine with an advanced query language is an effective and efficient solution as a foundation for the proposed precision medicine tasks. However, even though we created a framework to compose queries modularly, this proved to be a challenging task: building and tuning the queries and creating the reference standard still required deep medical knowledge and frequent iterations and tuning. Our results in the *biomedical articles* subtask (among the top 3 participants in all metrics) were significantly better than on the *clinical trials* subtask (below the top 10 in all metrics), correlating perfectly with the medical knowledge, iterations, and dedication that we devoted to each subtask.

Acknowledgments

We thank Marcus Bloice for proofreading the manuscript, and Lexigram, Inc. for providing us with an API key to access their medical knowledge graph. Our work is partially funded by the Brazilian National Research Council - CNPq (project number 206892/2014-4).

[11] https://github.com/Shugyousha/mesh2solrsyn

APPENDIX A: Index - Document Count and Fields

ALL BIOMEDICAL DATA - `ELASTIC_SEARCH/trec/_search?pretty`

```
"hits" : {
   "total" : 26739435,
    }
```

PUBMED RECORDS - `ELASTIC_SEARCH/trec/medline/_search?pretty`

```
"hits" : {
    "total" : 26669401,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "trec",
        "_type" : "medline",
        "_id" : "8381889",
        "_score" : 1.0,
        "_source" : {
          "pubmedId" : "8381889",
          "title" : "Characterization of acid-base transport mechanisms (...)",
          "publicationDate" : "Jan 1993",
          "abstract" : "RCCT-28A cells, a continuous cell line of rabbit (...)",
          "meshTags" : [
            "Acid-Base Equilibrium",
            "Animals",
            (...)
            "Proton-Translocating ATPases",
            "Rabbits"
          ]
        }
      }
      (...)
```

EXTRA ABSTRACTS - ELASTIC_SEARCH/trec/extra/_search?pretty


```
"hits" : {
    "total" : 70025,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "trec",
        "_type" : "extra",
        "_id" : "ASCO_86592-111",
        "_score" : 1.0,
        "_source" : {
          "pubmedId" : "ASCO_86592-111",
          "title" : "The INFORM HER2 Dual ISH assay compared with FISH (...)",
          "publicationDate" : "2011",
          "abstract" : "Background:  The HER2 gene, located on chromosome 17 (...)"
        }
      }

      (...)
```

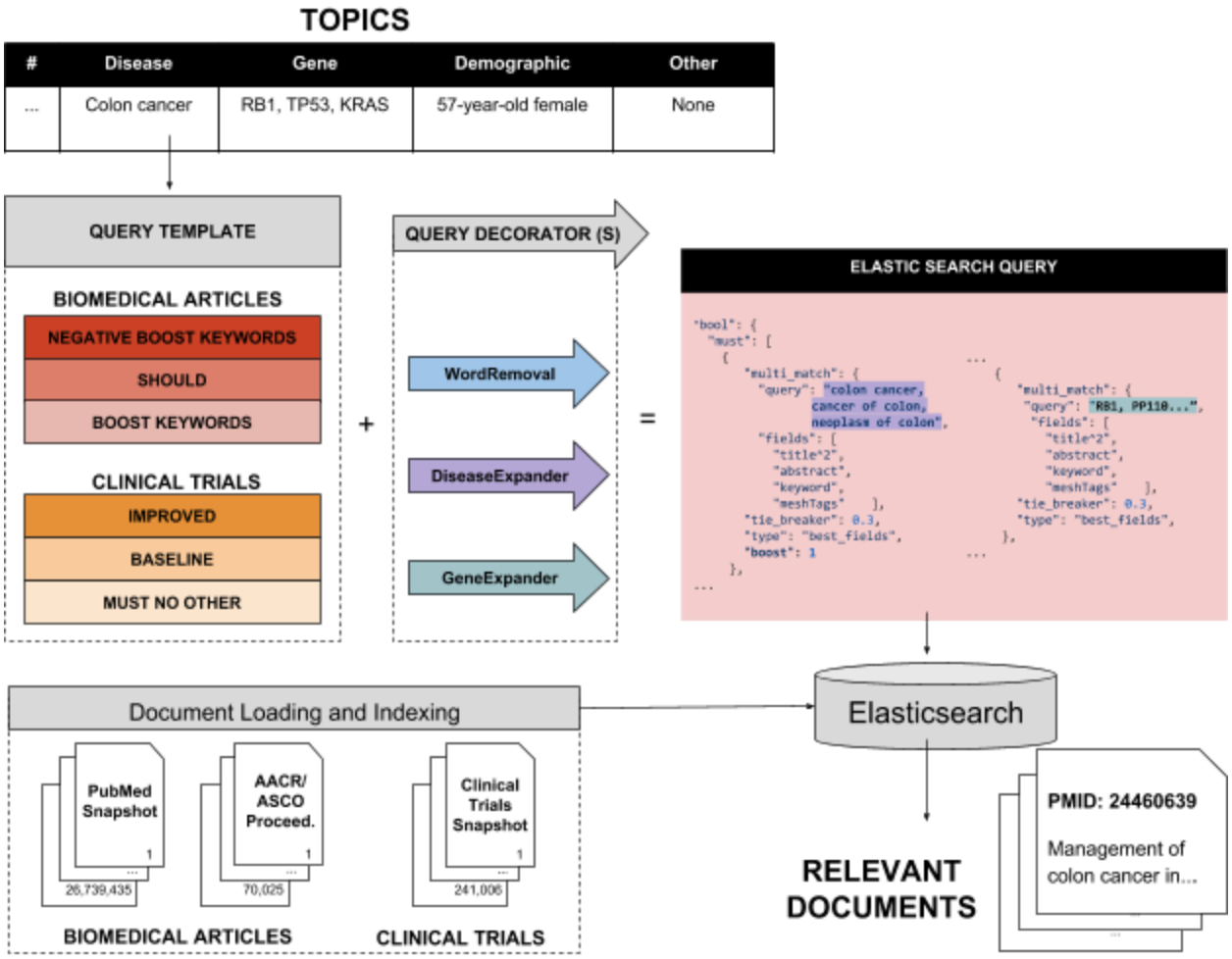
CLINICAL TRIALS - ELASTIC_SEARCH/clinicaltrials/_search?pretty


```
"hits" : {
    "total" : 241006,
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "clinicaltrials",
        "_type" : "clinicaltrials",
        "_id" : "NCT01097850",
        "_score" : 1.0,
        "_source" : {
          "id" : "NCT01097850",
          "title" : "Topical Henna Preparation for the Treatment (...)",
          "summary" : "The purpose of this study is to determine whether (...)",
          "sex" : [
            "female",
            "male"
          ],
          "minimum_age" : 18,
          "maximum_age" : 100,
          "inclusion" : " Patients receiving capecitabine and (...)",
          "exclusion" : " Patients with preexisting dermatological condition (...)"
        }
      }

      (...)
```

APPENDIX B: Framework, Query Templates, and Query Decorators

As our infrastructure for the experiments, we built a modular framework that could be improved and adapted for other TREC competitions, where final queries were built using blocks called query templates and query decorators. The framework is publicly available under the MIT license[12].

Our setup allowed us to quickly iterate on the continuous cycle of standard creation, evaluation, and system tuning. A general overview of the whole infrastructure is shown below.



Overview of the framework and infrastructure. Queries are built using modular blocks called *query templates* and *query decorators*.

[12] https://github.com/bst-mug/trec2017/

Query templates materialized valuable search strategies as a valid JSON Elasticsearch query and allowed for on-the-fly modification of topic dimensions (e.g. *disease*, *gene*). An excerpt of our *Negative Keywords Boost*[13] template is shown below, where `{{disease}}` and `{{gene}}` are placeholders.

QUERY TEMPLATE (excerpt of `negative-keywords-boost.json`)

```
{
    "bool": {
        "must": [
            {
                "multi_match": {
                    "query": "{{disease}}",
                    "fields": [
                        "title^2",
                        "abstract",
                        "keyword",
                        "meshTags"
                    ],
                    "tie_breaker": 0.3,
                    "type": "best_fields",
                    "boost": 1
                }
            },
            {
                "multi_match": {
                    "query": "{{gene}}",
                    "fields": [
                        "title^2",
                        "abstract",
                        "keyword",
                        "meshTags"
                    ],
                    "tie_breaker": 0.3,
                    "type": "best_fields"
                }
            }
        ],

        (...)
```

Query decorators dealt with the topic's dimensions before they were inserted in a template. For example, the *Disease Expander*[14] query decorator retrieved all known synonyms for the patient's disease using an external API call to a medical knowledge graph and inserted those in the `{{disease}}` placeholder of the template, instead of the original disease dimension from the topic. Query decorators were designed in a fluent way[15] so they could be concatenated.

[13]https://github.com/bst-mug/trec2017/blob/master/src/main/resources/templates/negative-boost-keywords.json
[14]https://github.com/bst-mug/trec2017/blob/master/src/main/java/at/medunigraz/imi/bst/trec/query/DiseaseExpanderQueryDecorator.java
[15]https://martinfowler.com/bliki/FluentInterface.html

QUERY DECORATOR (excerpt of `DiseaseExpanderQueryDecorator.java`)

```
public class DiseaseExpanderQueryDecorator extends QueryDecorator {

	public DiseaseExpanderQueryDecorator(Query decoratedQuery) {
		super(decoratedQuery);
	}

	@Override
	public List<Result> query(Topic topic) {
		expandDisease(topic);
		return decoratedQuery.query(topic);
	}

	private void expandDisease(Topic topic) {
		String disease = topic.getDisease();

		List<String> synonyms = null;
		try {
			synonyms = GraphUtils.addSynonymsFromBestConceptMatch(disease);
		} catch (UnirestException e) {
			e.printStackTrace();
			return;
		}

		String expandedDisease = String.join(" ", synonyms);
		topic.withDisease(expandedDisease);
	}

}
```

The code excerpt below shows the key snippets of how an experiment using the query template and decorator above was implemented:

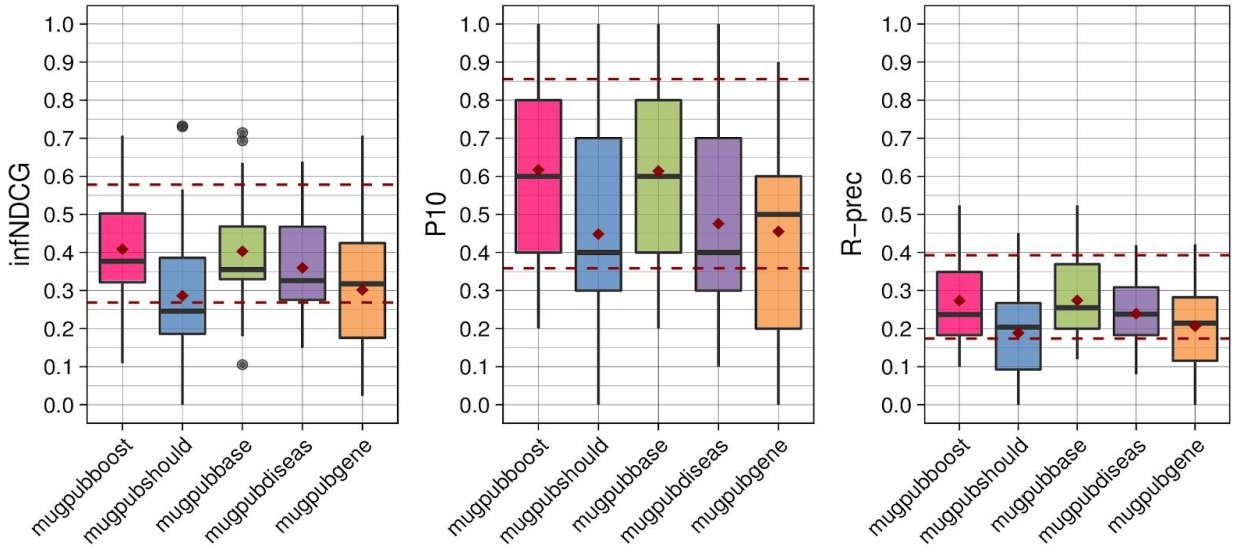
```
final File negativeBoostKeywordsTemplate = new File(
PubmedExperimenter.class.getResource("/templates/negative-boost-keywords.json")
.getFile());


public ExperimentsBuilder withDiseaseExpander() {
  Query previousDecorator = buildingExp.getDecorator();
  buildingExp.setDecorator(new DiseaseExpanderQueryDecorator(previousDecorator));
  return this;
}


builder.newExperiment().withTemplate(negativeBoostKeywordsTemplate).withDiseaseExpan
der();
```

APPENDIX C: Runs, Results, and Strategies

In this appendix we list the experiments that we finally submitted as runs and show the main strategy followed and final results according to the official evaluation metrics when we attended the TREC 2017 conference. Note that at that point of time one topic judgment was missing.

Biomedical Articles

STRATEGY/RUN	mugpubboost	mugpubshould	mugpubbase	mugpubdiseas	mugpubgene
Match disease & gene	must	should	must	must	must
Boost+ keywords	✔	✔	✔	✔	✔
Boost- keywords	✔	✔		✔	✔
Boost+ chemo. suffixes	✔	✔		✔	✔
Disease expansion				✔	
Gene expansion					✔
RESULTS					
infNDCG	0.4088	0.2864	0.4031	0.3596	0.3016
P@10	0.6172	0.4483	0.6138	0.4759	0.4552
R-Prec	0.2735	0.1887	0.2743	0.2393	0.2065



Clinical Trials

STRATEGY/RUN	mugctboost	mugctdisease	mugctbase	mugctgene	mugctmust
Match age range & sex	must	must	must	must	should
Match disease & gene	must	must	should	must	should
Comorbidities NOT excl.	✔	✔		✔	✔
Boost+ keywords	✔	✔		✔	
Disease expansion		✔			
Gene expansion				✔	
RESULTS					
P@5	0.2500	0.0929	0.3000	0.2357	0.2929
P@10	0.2286	0.0679	0.2643	0.2321	0.2536
P@15	0.1952	0.0738	0.2262	0.2119	0.2143