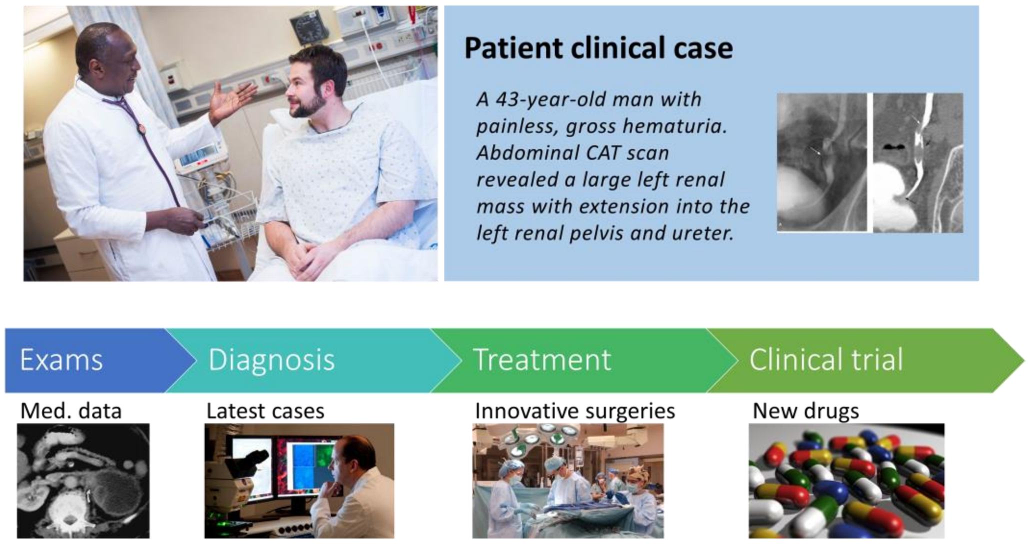
Matching
Patient Cases to Clinical Trials

Information Retrieval Course
2022/2023

Adapted from [TREC Clinical Trials TRACK]

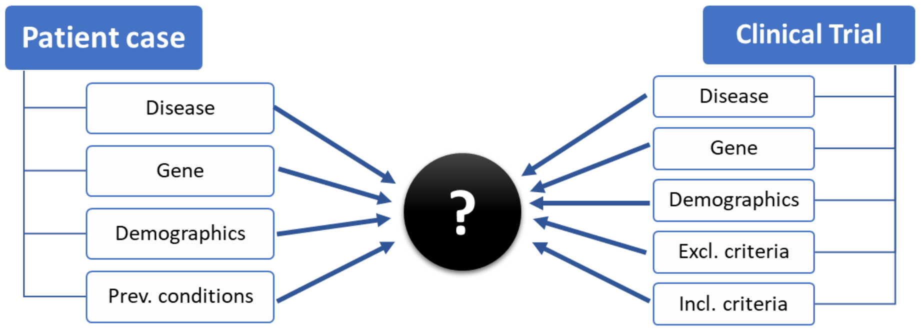


A 43-year-old man with painless, gross hematuria. Abdominal CAT scan revealed a large left renal mass with extension into the left renal pelvis and ureter.

The clinical decision process is many times supported by eHealth plaforms that help medics and other health professionals make the best decisions to their patients. The regular workflow is to decide the exam, to determine the causes and finally decide the treatment. In cases where the treatment is not successful, the last resort is a clinical trial where an new possible treatment is tested on patients.

Unfortunately, the vast majority of clinical trials fail to meet their patient recruitment goal. NIH has estimated that 80% of clinical trials fail to meet their patient recruitment timeline and, more critically, many (or most) fail to recruit the minimum number of patients to power the study as originally anticipated. Efficient patient trial recruitment is thus one of the major barriers to medical research, both delaying trials and forcing others to terminate entirely.

In this project you will build a system to retrieve clinical trials from ClinicalTrials.gov, a required registry for clinical trials in the United States. The goal is to find clinical trials where patients can be enrolled. The figure below provides a top-level illustration of the process: by examining the various enrollment criteria and the patient clinical record, a natural language + parameters matching algorithm is used to decide and rank which clinical trials are more likely to fit the patient case.



Requirements

- You should have a computer where you can run experiments and implement the required algorithms. The Google Colab is a good substitute for this setup.
- It is advisable to use an Anaconda Environment with the libraries identified in the provided YAML file.
- It is advisable to have a working JupyterLab and PyCharm installation.
- You're free to use the programming environment of your choice.

Methodology

Protocol

- Read and parse the clinical trial documents.
- Create an index structure for the different fields of the clinical trials
- Read and parse the queries, i.e., the patients case description
- Select a random sample of 80% patient cases to be your training set. The remaining 20% cases will be your test set.
- With the training patient cases, train or calibrate your *retrieval algorithms*.
- With the test patient cases, compute the top 100 candidate clinical trials using a *retrieval algorithm*.
- Measure the success of the proposed algorithm with the proposed metrics.

Dataset

Documents. Clinical trial descriptions can be quite long, but the core aspect of the trial description are the inclusion/exclusion criteria. These are not all-inclusive statements about the trial to the point that other trial information can be ignored, but they are key aspects to defining trial eligibility.

Queries. The queries of this system will be a lengthy (5-10 sentence) patient case description that simulates an admission statement in an EHR. The queries are limited to just the free text description of a patient record, as the structured data in EHRs, while helpful, is outside the scope of this project.

Metrics

To assess the retrieved documents, you will use both system utility metrics and system stability metrics.

For system utility metrics, you should use:

- **Precision@10** to measure the percentage number of correct documents in the top 10 results.
- **nDCG@5** to measure the cumulative gain of retrieving multi-level relevance documents on the top 5 results.
- **Recall@100** to measure how many relevant results are not accessible to users.

For system stability metrics, you should use:

- **Mean Average Precision** to measure the robustness of the compute ranks,
- **Precision-recall curves** are an informative visualization of the performance of the system across the entire rank of results.

The evaluation will further be broken down into *eligible*, *excludes*, and *not relevant* to allow retrieval methods to distinguish between patients that do not have sufficient information to qualify for the trial (*not relevant*) and those that are explicitly excluded (*excludes*).

Phase 1: Base Pipeline

Deadline: 13 Oct

The clinical trials corpus and the code snippets to load the data are available here:

https://drive.google.com/drive/folders/17-h0XQyGKED7trvarR38c1OWh-9UypHl?usp=sharing

Reading the Clinical Trials Documents

Examine the code provided to read the documents. Inspect the different sections of a clinical trial document. **Clinical trials will be your corpus of documents.**

Due to the large size of the clinical trials database, you should only load the clinical trials that are in the `qrels-clinical_trials.txt` file. The documents outside that file were not judged by assessors, hence, we don't know if they are relevant or not.

We provided you with a parser to read the contents of a compressed file (without decompressing it). However, if wish to extract the contents of the file, note that it will occupy ~3GB in your disk and because it will extract > 100k files it may freeze your computer when you access the folder with the clinical trials.

Reading the Patient Cases

Examine the code provided to read the patient cases. **Patient cases will be your queries.**

Vector Space Model

Using the scikit-learn implementation of the VSM with TF-IDF weights and cosine distance, build an index for the entire set of documents for which you have ground-truth. Consider the example below.

```
from sklearn.metrics.pairwise import pairwise_distances
from sklearn.feature_extraction.text import TfidfVectorizer

corpus = [
     'This is the first document.',
     'This document is the second document.',
     'And this is the third one.',
     'Is this the first document?',
 ]

# Learn a vocabulary of unigrams and bigrams
index = TfidfVectorizer(ngram_range=(1,2), analyzer='word', stop_words = None)
index.fit(corpus)

# Compute the corpus representation
X = index.transform(corpus)

# Compute the query representation
query = ['document']
query_tfidf = index.transform(query)

# Compute the query-corpus similarity for all documents
doc_scores = 1 - pairwise_distances(X, query_tfidf, metric='cosine')
print(doc_scores)
```

Use only the `brief_title` section of the clinical trial documents.

Language Models

Implement the Language Model with Jelinek-Mercer Smoothing. To access the corpus statistics you can use scikit learn's CountVectorizer class.

Evaluation

Implement an experimental test-bed to select the algorithm that best solves the problem:

- Examine the code provided to compute experimental results.
- Prepare a retrieval test-bed by dividing your queries into a training set and a test set. The corpus will be static.
- You have ~60 test/training patients (queries), and for each patient (query) you need to order the trials that best match the patients cases (the queries).
- This retrieval test-bed needs to be organized in a way that you run experiments systematically and are able to keep adding new models to the testbed.
- Compute all the provided metrics to evaluate the obtained results.
- Organize your results into tables and graphs to support your discussion. We suggest you to use wandb or matplotlib to visualize your data.

Report

The project report should have the following structure:

- Introduction
- Implemented methods
- Experimental setup
- Results discussion
- Conclusions

You will be writing your project report incrementally. On each phase you will be adding 4 pages to your report with new information.

Your report must follow one these templates:

- Word: https://www.springer.com/gp/authors-editors/journal-author/word-template-zip-154-kb-/22044
- Latex: https://www.overleaf.com/latex/templates/springer-lecture-notes-in-computer-science/kzwwpvhwnvfj

Suggestions

1. **Pickle:** To save time you can use pickle to save and load variables to disk files:

   https://wiki.python.org/moin/UsingPickle