**Gonçalo Carmo de Araújo**

Degree in Computer Science

# Biomedical information extraction for matching patients to clinical trials

Dissertation for obtaining the Degree of
Master in Computer Science

Adviser: João Magalhães, Assistant Professor,
Univsersidade Nova de Lisboa/FCT

Co-adviser: André Mourão, Researcher,
Univsersidade Nova de Lisboa/FCT

:

President: Dr. Ana Maria Diniz Moreira

Others: Dr. João Carlos Amaro Ferreira

Dr. João Manuel da Costa Magalhães

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

**September, 2018**

**Biomedical information extraction for matching patients to clinical trials**

*For those who seek knowledge, the only true path to happiness and fulfilment.*

# Acknowledgements

I would first like to thank my thesis advisor João Magalhães of the Faculdade de Ciências e Tecnologias at Univsersidade Nova de Lisboa. Who was always there for any question, or difficulty i ran into. Helping me develop my work my own way, while always steering me into the right direction. I would also like to thank my co-adviser André Mourão of the Faculdade de Ciências e Tecnologias at Univsersidade Nova de Lisboa. While finishing his PhD, he still had time to ear some ideas i had and give me multiple writing advises. A special Thanks to all my colleagues from the NovaSearch laboratory.

One big thank you to Faculdade de Ciências e Tecnologias and all the wonderful people i met there, who shared this road with me. Helping me and growing alongside, couldn't have asked for better friends and colleagues. This thesis is also dedicated to my friends outside the university, that had to put up with a lot of conversations ending up in machine learning and why retrieval system can revolutionize the world around us. For both, thank you for been truth to me and yourselves doesn't matter the time and circumstances, always following the righteous path.

Thank you my beautiful family for this 24 years of happiness and understanding, it wouldn't be the same without you, each one of you with no exception, to my Aunts and uncles, Grandmother and Grandfather. One special thanks to my aunt Carolina, who was always there for any advise i had while doing this thesis and in life.

Last but never ever least, thank you to my three favorite people in the whole world, my brother, father and mother. Thank you for supporting me in all ways possible and yes i do include the financial part on this many possible ways, well at least I'm always smiling i guess its like tipping that very nice waiter, it's fine! For real thank you for putting up with me when i wake up thinking the world owe me something, showing me my rights and wrongs, and sticking up for me no matter what. its a pleasure to be one fourth of this beautiful team. Love you all my friends and family, give thanks and praise to this new chapter that's opening and not the one that just finished.

# Abstract

Digital Medical information had an astonishing growth on the last decades, driven by an unprecedented number of medical writers, which lead to a complete revolution in what and how much information is available to the health professionals.

The problem with this wave of information is that performing a precise selection of the information retrieved by medical information repositories is very exhaustive and time consuming for physicians. This is one of the biggest challenges for physicians with the new digital era: how to reduce the time spent finding the perfect matching document for a patient (e.g. intervention articles, clinical trial, prescriptions).

Precision Medicine (PM) 2017 is the track by the Text REtrieval Conference (TREC), that is focused on this type of challenges exclusively for oncology. Using a dataset with a large amount of clinical trials, this track is a good real life example on how information retrieval solutions can be used to solve this types of problems. This track can be a very good starting point for applying information extraction and retrieval methods, in a very complex domain.

The purpose of this thesis is to improve a system designed by the NovaSearch team for TREC PM 2017 Clinical Trials task, which got ranked on the top-5 systems of 2017. The NovaSearch team also participated on the 2018 track and got a 15% increase on precision compared to the 2017 one. It was used multiple IR techniques for information extraction and processing of data, including rank fusion, query expansion (e.g. Pseudo relevance feedback, Mesh terms expansion) and experiments with Learning to Rank (LETOR) algorithms. Our goal is to retrieve the best possible set of trials for a given patient, using precise documents filters to exclude the unwanted clinical trials. This work can open doors in what can be done for searching and perceiving the criteria to exclude or include the trials, helping physicians even on the more complex and difficult information retrieval tasks.

**Keywords:** Medical Text Retrieval; Query expansion; Information Retrieval ; Rank Fusion; Information Extraction;

# RESUMO

A informação médica no formato digital teve um crescimento estonteante nas últimas décadas, o que desencadeou uma revolução total em como e quanta informação está disponível para os profissionais de saúde.

O problema desta onda de informação prende-se com o facto de se tornar um trabalho extremamente exaustivo para os profissionais de saúde, fazer uma selecão precisa da informação espalhada pelos repositórios médicos. Este é um dos maiores desafios para os médicos nesta nova "Era" digital: Como reduzir o tempo despendido a ler e a selecionar documentos para concluir formas de criar uma correlação entre os pacientes e estes (intervenções médicas, clinical trial, receitas médias, etc.) .

O Precision Medicine 2017 é um desafio proposto pela Text REtrieval Conference (TREC), em que o foco principal é a resolução de problemas como os descritos anteriormente, no âmbito da oncologia. Usando um dataset que devido ao elevado número de clinical trials, está muito perto dos casos reais onde sistemas de information retrieval são implementados em hospitais ou clinicas. Este desafio é um bom ponto de partido para aplicar information extraction e métodos de information retrieval, num dominio tão complexo como é a oncologia.

O objetivo deste trabalho é melhorar o sistema que foi previamente desenvolvido e submetido para o dataset de clinical trials do TREC PM 2017. Trabalho este que ficou no top-5 dos sistemas de 2017. A equipa da NovaSearch baseando-se no trabalho desenvolvido nesta Tese conseeguiu um aumento de 15% de precisão no dataset de 2018, quando comparado com 2017. Para o desenvolvimento deste trabalho foram utilizadas várias técnicas de IR para a extração e processamento de informação relevante, incluindo rank fusion, Query expansion (Pseudo relevance feedback, Mesh terms expansion, etc.) e experiências com algoritmos de Learning to Rank (LETOR). O objectivo é obter para cada paciente o melhor conjunto possível de clinical trials que possam ajudar a curar o paciente, usando filtros para excluir os documentos indejesados e para dar mais relevância a outros. Este trabalho tem o potencial de abrir portas no que diz respeito ao que pode ser feito para melhorar a pesquisa e a interpretação dos criterios de exclusão e inclusão dos documentos, ajudando os profissionais de saúde nos desafios mais complexos de obtenção e extração de informação das bases de dados médicas

**Palavras-chave:** Medical Text Retrieval; Query expansion; Information Retrieval ; Rank Fusion; Information Extraction;...

# Contents

# List of Figures

# List of Tables

# LISTINGS

CHAPTER 1

# INTRODUCTION

## 1.1 Context and Motivation

The towering amount of biomedical data accessible to physicians is one of their most helpful resources. Most of the medical world information is available on search engines as PubMed and Clinicaltrials.gov (Figure 1.1), where users can search for almost any kind of medical information. Among physicians, digital literature is now the most popular way to either read or publish articles, all throughout the world this new digital era is becoming the standard. The work developed by [64], clearly shows how this digital growth has influenced positively the medical panorama on China, a country with a record growth both technologically and economically.

The digital medical literature gives users the opportunity to create documents not only with narrative notes, but also with multiple ontology references which help others on their synthesis work. MeSH[1] terms as "Neoplasm" and "cancer" are an example of ontology references, that can be used to help on the retrieval of documents. As the digital literature evolves and grows in numbers, a lot of new challenges emerge: having a gigantic database of articles and references can be a problem if there is no simple and effective way to query the dataset for very precise answers.

The complexity of certain diseases and the differences in opinion between physicians regarding correct treatments makes the evaluation of solutions very complex and time-consuming. The oncological diseases are a good example, because for the same cancer different patients could have multiple reactions. These assumptions can be extracted from the examples of relevance documents for each query, provided by the TREC PM 2017 organization.

Misjudging this disorders can be the difference between life and death. Since the

[1] Medical related vocabulary

number of medical literature in collections like Medline[2] can reach 27 millions articles, manually sifting through all these documents to structure a treatment is not feasible. So the goal of this thesis is to create a system that can rank clinical trials according to their relevance to a query (i.e. Clinical Report of a patient) made by a physician. Such systems can help finding candidate patients to on-going or future clinical trials, saving physicians the hardship of manually performing this selection.

TREC (Text REtrieval Conference) Precision Medicine Track aims to provide clinicians with important information to support the medical decision process, and to help them avoid choosing solutions that can be potentially armful for the patients (e.g. not recommending trials that explicitly exclude patient with diabetes for a diabetic patient). The "Precision medicine" paradigm is based on the assumption that domain specific techniques are needed in some cases, as expanding the terms used on a query with Medical Subject Headings (MeSH) terms to help matching the query to MedLine articles (manually annotated with MeSH terms). Existing systems can be improved to fulfill this need, by adding such to the retrieval process, beside applying this techniques this work extends them not only to articles but for clinical trials. This was one of the main insights from organizers of TREC Precision Medicine 2017. They found that systems with specific domain information achieved better performance that generic retrieval systems [56].

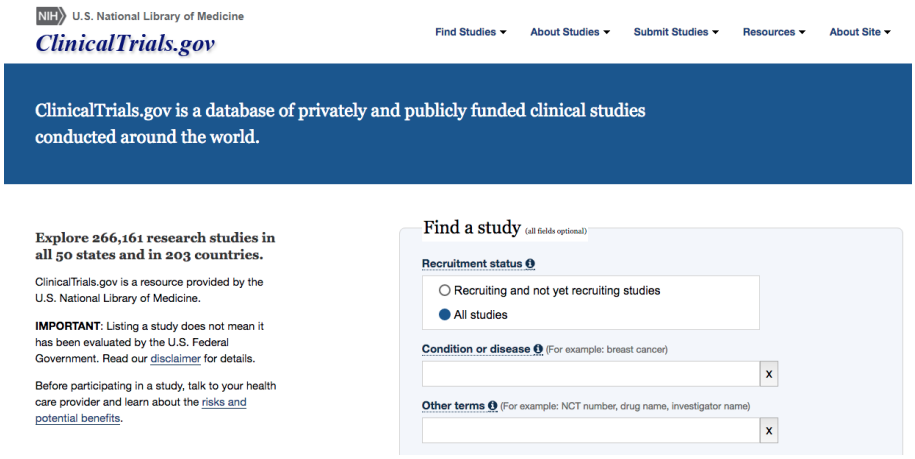


Figure 1.1: clinicaltrials.gov official site initial page, with search for disease in clinical trials at clinicaltrials.gov database.

## 1.2 Objectives

The purpose of this work is to understand how information retrieval (IR) methods applied with Natural Language Processing (NLP) and Machine learning (ML) techniques can improve medical IR, specially for cases when a high level of precision is required. More specifically, the system is going to be focused on oncology cases, characterized for its specificity regarding each individual person type of cancer and genes mutation. So the

[2]bibliographic database of life sciences and biomedical information.

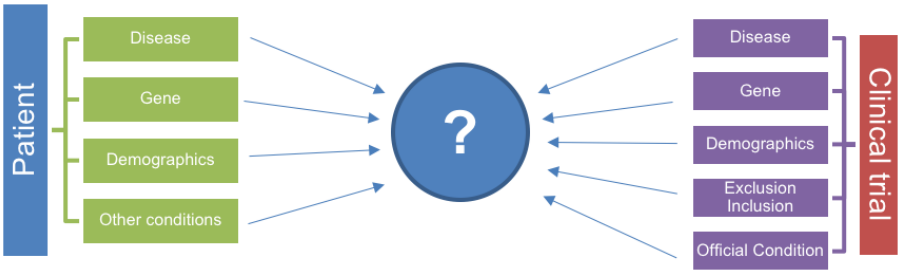


Figure 1.2: Input data used for the development of this retrieval system.

developed methods scrutinize the most important aspects present on clinical trials and patients information, to perceive how relevant a given document is for a specific patient. For example, using domain specific Named Entity Recognition techniques to find genes in clinical trail text, and filtering out the ones that do not match the patient gene mutation.

At the core of this work was the TREC PM 2017 Clinical Trials dataset. It contains structured clinical trials from oncological interventions and 28 patient cases, shown on Table A.4. Since the query topics for TREC PM 2017 are all oncology cases (i.e. including patient demographics, type of cancer and specific gene mutation), it is necessary to retrieve the documents as precisely as possible so that the IR system can only retrieve trials fitting both gene and disease information. So having a dataset divided into narrative fields (e.g. textual descriptions) and non-narrative fields (e.g. fields that contain very specific information like age, gender or primary purpose of the trial) enables us to reach not only relevant documents on the topic, but also filter those that are highly relevant to the topic but cannot be applied to that specific medical case (e.g. patient does not fit the trial requirements). To create the retrieval system, multiple NLP methods were applied on the narrative descriptions, enabling the IR system to match queries and documents. In addition to NLP techniques, it was also used multiple retrieval functions, and it was also applied machine learning techniques in order to learn new relations between data from its structure. For example using decision trees algorithms to perform ordinal regression on the clinical trials, based on the information extracted with the developed IR methods. This work is based on [4], which details NovaSearch work group submission for TREC PM 2017, where we applied some of the previously described techniques.

To improve retrieval results IR techniques such as, Pseudo Relevance Feedback were used to perform Query Expansion, by adding to the query new terms that may be relevant to the medical case. Rank fusion is another set of techniques used on this work, which consists in combining the results of different retrieval algorithms and techniques for the same query. In this work, we have used two Rank Fusion algorithms: Reciprocal Rank Fusion (RRF) and CombSUM Rank Fusion.

The final result is a clinical decision support system, capable of understanding medical clinical trials and make correlations between this data and patients conditions, focused entirely on cancer patients (figure 1.2). The idea of the system is to retrieve relevant documents for a specific cancer in a given patient. Utilizing multiple information retrieval

methods, created by applying natural language processing and information extraction principles. And it was also used the machine learning algorithms based on these information retrieval methods. System's quality was ensured by TREC PM 2017 relevance judgment.

## 1.3 Thesis Structure

This remainder of the thesis is subdivided in 6 chapters:

- **Chapter 2:** describes background and related work techniques.
- **Chapter 3:** gives insights on basic oncology fundamentals.
- **Chapter 4:** explains how the benchmarks for this work were created (Indexing, Retrieval, Query Expansion, Re-ranking)
- **Chapter 5:** explains how machine learning task, learning to rank (LETOR), was used on this work.
- **Chapter 6:** describes the Evaluation process and retrieval results.
- **Chapter 7:** details the future work that could be done to improve the system.

CHAPTER 2

# ONCOLOGY DOMAIN

This chapter was created with the purpose of providing readers with some basic knowledge, on the oncology domain. The goal is to provide the tools to perform a quick and simple parallelism between, the oncology field of medicine and some of the options taken during the system implementation. Studying domain specific information helped me on understanding the theoretical effectiveness of certain domain assumptions, which were later applied to the system on the following chapters (4 and 5) results.

## 2.1 Introduction to Cancer diseases

Cancer is a disorder on cells, and although the term is commonly (and correctly) associated to tumor, this mass of cells only represent the final stage of cancer. Changes on cells take years to reach a tumor. These cell changes are the real classifiers of cancers: different types of changes are created by different gene mutations on different body areas, which will end up on very different cancer diseases. Cancer, as described on [38], consists on a continuous abnormal response to the cells control mechanisms. This disorders affect two important mechanisms: cell growth (size of individual cell) and proliferation of cells.

Normal cells follow a cycle for pre-defined growth, proliferation and death scenarios. For example, if an injury occurs on a tissue, some of the cells that make up this tissue are programmed by these control mechanisms to grow and proliferate on higher rates until the tissue is fully reestablish. When the process of healing is completed all cells return to their normal life cycle, balancing the cells count and production on that tissue. What happens during the formation of cancers is exactly the opposite: cells stop following normal control mechanisms which could cause abnormal proliferation, or abnormal growth of cells because cells stop dividing. These small changes could lead to the creation of dense masses of modified cells, the so called tumors.

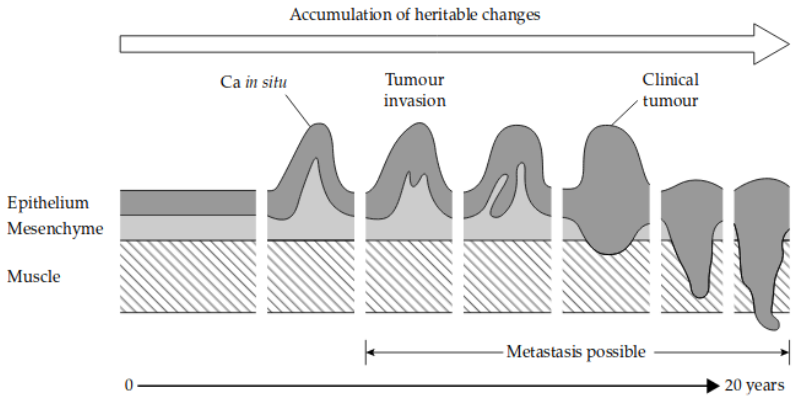


Figure 2.1: All fundamental stages of cancer development. This image was extracted from [38].

### 2.1.1 Types of tumors

The process of cancer development (carcinogenesis), is the sum of multiple abnormal cells within a tissue, which is known by the name of dysplasia the actual condition the patient suffers when the cancer is developing [38, 60]. The severity of the tumor is correlated to the stage of dysplasia, and their invasive capability. Figure 2.1 shows the difference between multiple stages of the cancer development.

Given this two parameters (dysplasia and invasive capability), according to [38] tumors can be classified as following.

**Benign Tumor:** Shows the initial stages of dysplasia, already with a considerable cell count of abnormal types. This types of tumors are the least serious, as the common form of tumor is not yet perceptible, there is no significant mass change, and no type of tissue invasion is possible.

**In situ Tumor** Also called pre-cancer, is the second stage of cancer development. On this tumors, the dysplasia becomes a serious issue, leading to the growth of cell masses that become noticeable. On this stage, there is no invasion of blood system or neighbouring cells, and thus it is not technically classified as cancer.

**Malignant Tumor** The final stage of cancer, characterized by the existence of big masses of cells pressuring and invading neighbouring tissues and vessels. On this stage, the cancer reaches very high proliferation rates, that tend to increase as the cancer grows. This types of tumors also called "Solid tumor" or "Solid neoplasm" are the main focus of this work, since clinical trials are last resource procedures applied to very specific clinical cases.

## 2.2 Diagnosis and relevant factors to cancer diseases

Cancer diseases as explained previously have multiple factors that influence the carcinogenesis (i.e. process of cancer development), but what is really relevant for diagnosis and following treatment is the location of the cancer and gene mutation responsible for cell changes. Since cancers are developed by genes that have been expressed inappropriately or mutated.

To correctly diagnose a specific cancer and thus, understand which actions should be taken, one should know what gene mutations are in action. After detecting the genes, it is necessary to characterize all relevant mutations for the domain, according to their function for the carcinogenesis. Some genes may be the initiators, others developers and others are responsibly for the maintenance of the cancer. All genes can perform one of these previous functions, and depending on the tumor stage some types of mutations must be prioritized over others. So, it is mandatory to understand what genes are doing what on the tumor development process. With this information well defined, it becomes much more simple for physicians to create groups of patients with the same genes and disease, leading to much more controlled environments for this therapies. It is important to understand that cancer therapies have come a long way, but still have a long way to go. Having this controlled environments is very important for helping improve individual therapies overall survival rate.

In terms of the work developed, these diagnosis notions will be used as the base for our queries, using both disease and gene mutations extracted from a single patient to query the database of clinical trials. The goal is not only to use the exact disease and gene text but deriving queries from these texts, which will be explained on the following section.

## 2.3 Treatments and Clinical trials enrollment

According to the National Cancer Institute (USA) [35], from 1990 to 2014 the overall death rate of cancer in the U.S.A. dropped some astonishing 25%. However, the institute estimates that around 600.000 patients will die from cancer only on 2018. This huge number is the main reason why a multi-area and multi-country effort is being made to fight this group of diseases.

Since cancer treatments are still a work in progress and drastically vary from patient to patient, they are enrolled in clinical trials for procedures and medications that were previously tested in animals. Figure 2.2 shows some of the basic ideas to differentiate clinical trials and which patients should be part of it. But in order to assign patients to clinical trials that will not harm and possibly help the patient, clinical trials must be filtered according to cancer treatments assumptions and the patients actual disease. First, as explained on [38], as clinical trials exist not only for cancer diseases, these must be of an interventional type and not an observation. Only interventional trials attempt to treat the cancer. Beside that only a small range of intervention types are expected to be performed,

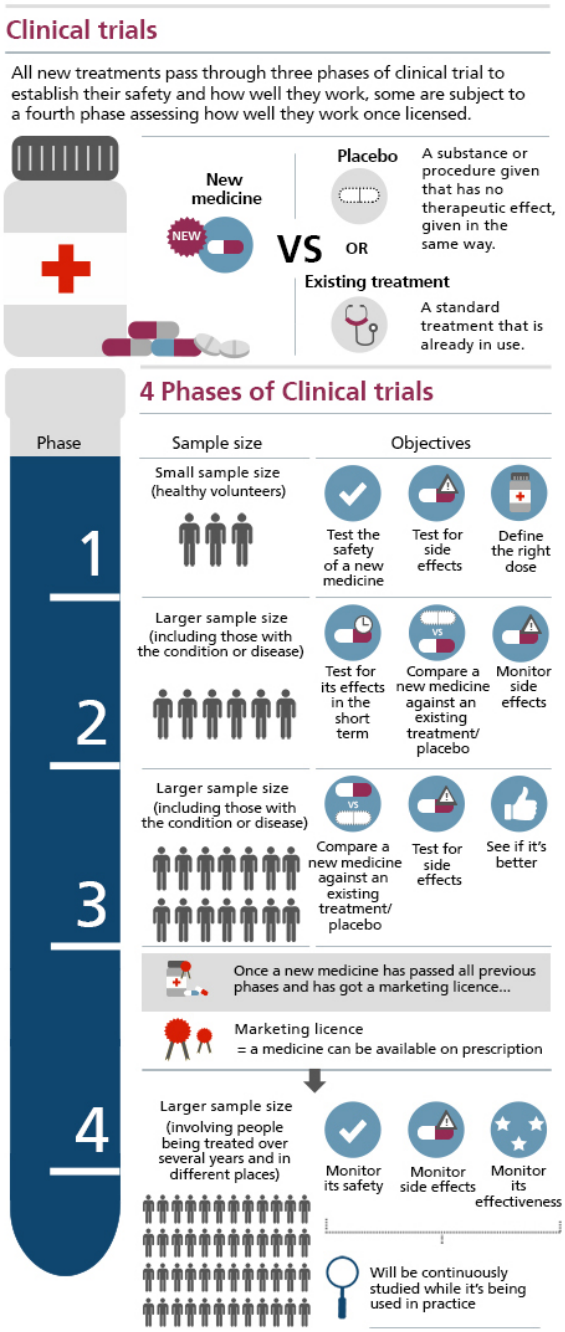


Figure 2.2: The clinical trial workflow, extracted from [65].

either: Chemotherapy, radiotherapy or biological medication use.

On the other hand, patients information is very important to help exclude undesired trials for that specific person. Features such as their gender and age are essential for physicians to match patients and trials. Demographic information is viewed as one of the most relevant features for cancer treatments throughout medical literature [38]. As said on [38], age is definitely one of the biggest risk factors, since most of the cancers take a few years to developed and should normally occur on older population.

Clinical trials have disease and genes explicitly expressed as conditions, inclusion criteria, or as keywords. That could be automatic keywords (from MeSH and SNOMed), or manual keywords introduced by the clinical trial author. These fields on clinical trials have a considerable level of accuracy and can be very helpful for clinical trials readers to match between clinical trials focus and the patient needs. On the other hand, the expressed exclusion criteria is used for exclude patients from trials, also based on their gene, disease and other conditions the patient may suffer from. For more detail on how this oncology assumptions can be used for information retrieval, and how they were implemented on this work go to Chapter 3 and Chapter 4 respectively.

CHAPTER 3

# BACKGROUND AND RELATED WORK

Several information retrieval systems have been created for computing the extraction of documents information and consequently retrieve and rank them from most to least relevant, as [50] were a clinical decision support system was created. These systems were designed to address the continuous growth of medical related information and the need to access it in a simple and quick fashion. This section describes some of the more commonly used techniques and their performance on existing retrieval systems.

## 3.1 Information Extraction

Information Processing is indispensable for information retrieval systems. It enables comparing documents and queries, making documents easier to manage for the retrieval process. It can be focused on two major ideas: processing structured text and unstructured text. Information Extraction (IE) is a computer science task, focused on processing unstructured data. IE goal is to find and give structure to relevant information (i.e. medical terms) on unstructured data (i.e. free text). For this purpose a lot of different approaches had been used in the midst of natural language processing: application of text parsing filters, IE detailed sub tasks(e.g. named entity recognition, relation extraction) already developed from NLP methods or information extraction based on domain specific taxonomies and ontologies.

### 3.1.1 Text parsing

NLP on its core consists on methods used for processing and understanding text written by and for humans, enabling system matching between blocks of unstructured text, as described by [8]. This subsection is entirely dedicated to some of the base methods that could be used for information extraction with NLP. All of them capable of processing

chunks of text in a pre-defined fashion, aiding matching between indexed information and query information. The primary idea of these methods is both to emphasize relevant pieces of information, while at same time non relevant textual information is discarded or shown in a way easy to perceive as irrelevant.

If we analyze the phrase, "the patient suffers from a Meningioma with already shown AKT1 mutation.", what is really necessary for the system is the disease and gene references. So the processed text should look something like "patient suffers Meningioma AKT1 mutation", removing most of the irrelevant information. The following are commonly used NLP techniques, extracted from the previous cited book and the following published books: [46], [15] and by [48].

**Tokenization:** Used to remove all form of punctuation and split text into little chunks of text, normally separated by white space or punctuation, called tokens.

**Lower case filter:** Converts all words into lower case.

**Stop word removal:** Remove common words like "this", "a", "or", that will occur in most of the English literature.

**Word grams:** This filter creates tokens from other tokens. For example, considering a minimum of 2 and a maximum of 3 for the size of tokens for each word gram, we get the following word grams for the sentence *"Pancreatic ductal adenocarcinoma"*:

```
Pancreatic ductal
ductal adenocarcinoma
Pancreatic ductal adenocarcinoma
```

This way terms as "Pancreatic ductal neoplasm" that previously wouldn't be matched, will be matched because of the word gram "Pancreatic ductal" which exist on both word grams lists.

**Character grams:** Creates n-grams of words, with a minimum and a maximum length for the words is given as a parameter. This is a technique that has been been successfully applied in the medical domain due to the complex spelling of medical terms (many common prefixes and suffixes). For example, using 3 as a minimum, 5 as maximum, in the sentence *"Solid tumor"*, we would have the following n-grams: `sol, soli, solid, oli, olid, lid, tum, tumo, tumor, umo, umor, mor`.

**Stemming:** Stemming is a process for reducing words to their word stem. For example: *cancerous*, *cancers*, and *cancer* would all be reduced to their root word *cancer*.

**Part-of-speech tagging:** The POS tagging consists in marking each word in the text with a specific grammatical group (e.g. nouns, adverbs, verbs). This techniques takes into account the context in which the word appear on the text, to correctly infer the grammatical meaning. A good example of POS usage is Stanford Part-Of-Speech Tagger developed on [62].

NLP can be very useful for scientific information retrieval. [21] shows how these techniques can be very useful for IR in the biology domain, where these techniques are applied at two essential points for IR. First at indexing, as said before and given that biology articles tend to be very extensive, NLP techniques removes irrelevant information and subsequently reduces the number of indexed terms, and is even a good practice when used to populate a database. Second it helps at searching time, applying exactly the same transformations on queries and helping on matching biology related information to either index or databases.

### 3.1.2 Named Entity Recognition (NER)

Named Entity Recognition is a sub-task of Information Extraction, which consists on the recognition of named entities within the text. These named entities are descriptors of terms or agglomeration of terms on the narrative. The challenges within NER can be broken-down into two problems: identifying the boundaries of a entity (identifying the terms that make up an entity), and the type of entity (e.g. Location, Person, Organization). NER can be applied to very different domains so for each domain, specific descriptors were designed. The framework Apache cTakes (referred on Section 3.6) is a good example of NER applied to the medical domain, including components as "Drug Named Entity Recognition" where Drug related descriptors are extracted. To perform this recognition multiple NLP techniques can be used.

### 3.1.3 Relation Extraction (RE)

Another sub-task of information extraction is Relation extraction, that is the next natural step after performing NER. It is commonly perceived as the prelude of Relation extraction, where the already named entities will be used to understand the relationship between them. One simple way of performing RE is with a set of two entities of types: A and B, and a pattern "s". Always on the same form (A, s, B). So for instance if entity A is described by "PERSON", entity B is described by "LOCATION" and the pattern "s" is equal to "lives in", we can automatically get the reference of persons that live on a certain location for every narrative already processed with NER.

### 3.1.4 Domain Taxonomies and Ontologies

The domain resources represented are predefined terminologies for the medical domain. These resources are used to create a consensual medical information representation across medical literature, and simplify the process of storing and finding relevant information. Information Retrieval systems can benefit greatly from these terminologies.

Medical Subject Headings (MeSH) [41] is a hierarchically-organized vocabulary, used for indexing and cataloging of medical information on databases as PubMed. These headings can be used in multiple ways but ideally they are automatically or manually annexed

to medical literature to serve as keywords at searching time. This way, Information retrieval systems tasks can be simplified when the documents being inspected discriminate these headings. This headings provide systems with a quick and easy way of perceiving not only the specific set of diseases detailed on the documents, but also possible interventions or drug use specifications, that could be unnoticed when performing text parsing. The following are examples of MeSH terms found on a clinical trial, from the TREC PM 2017 dataset, that is highly relevant for a person with a liposarcoma in which a CDK4 inhibitor was already administrated: e.g. "Liposarcoma", "Palbociclib". Where liposarcoma is the disease aimed on the trial, and palbociclib is a type of drug treatment intended to inhibit CDK4/6 gene mutations.

SNOMED Clinical Terms (CT) is a clinical health terminology, used to detail several types of medical information (e.g. diseases, conditions, interventions) in a very comprehensive and precise manner [27]. It uses a hierarchically structured, predefined terminology in order to detail medical information with the minimum noise possible. For example, using the query: "Retrieve the findings which have a benign tumor morphology". The SNOMED CT associated with this query are "benign" and "neoplasm", where the word tumor was changed for the more precise term for it on SNOMED CT database: "neoplasm". While all the remaining text was removed beside "benign". Since the documents to retrieve also have SNOMED CT, all of those who contain the terms "benign" and "neoplasm" are retrieved.

On Information retrieval, these terms can help improve the systems precision matching both document indexed with these taxonomies and queries extended also with them. Query expansion provides systems with a wide range of keywords for matching both queries and documents. Meanwhile the efficiency of systems using this taxonomies is co-related to the level of accuracy automatically or manually select keywords have, to each individual query or document.

## 3.2 Information Retrieval

### 3.2.1 Indexing

Indexing of information is one of Information Retrieval primitives. Consists in parsing and storing documents on a structure designed for efficient inspection. This structure is called an index, that can be interpreted as a dictionary of terms or documents. Indexing techniques are more effective when combined with NLP methods such as: splitting terms, normalizing terms, removing unwanted terms for search and many more methods. Creating text analyzers responsible for documents parsing and query parsing.

For term-based search the most used and efficient type is the inverted index, in which each term on the collection gets indexed with a list of documents that contain it. Figure 3.1 shows the creation process of an inverted index. Both query text and documents text are processed by the same analyzer, which will increase the matching possibility between

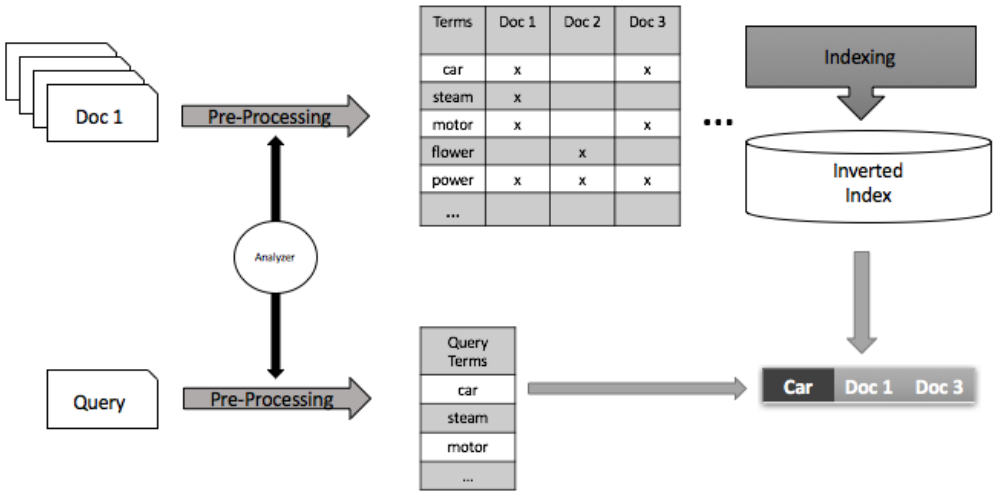


Figure 3.1: Inverted index creation methodology.

them. Terms and the references of the documents where each terms appear are indexed. At search time, the query terms are matched to the document terms, and the documents were each term appear are retrieved. To infer which documents are best for the query, retrieval models are used which are going to be explained on section 3.2.2.

### 3.2.2 Retrieval Models

These models (also called ranking/retrieval functions) are responsible for ranking the documents according to each one individual extracted terms. Retrieval functions are the core of the IR systems; they calculate the similarity between query and documents and rank these documents by their relevance to the query. Various models were created using different assumptions on how to score the document according to the extracted information. Some models are based on geometrical/linear spaces, as the TF-IDF model, while other models use probabilistic ranking models or Language models (i.e. based on probabilistic models but also taking into account the context in which the terms appear as LMD explained bellow). This section is based on techniques described on two key Information Retrieval books, [46] and [15].

**Term frequency-Inverse document frequency (TF-IDF):** Statistical method to define the importance of words for a specific document and simultaneously for the collection. If a word frequency in a collection is low, and its frequency in a document is high, its TF-IDF value is higher than the value for words that occur more often in the collection, even if they have high frequency in a few documents. This model has two functions: Term frequency, as the name suggests is the total frequency of a term within a document. Inverse document frequency, calculates how relevant a given term is within the entire collection.

The following are the TF and IDF equations:

$$TF_i(d) = \frac{ni(d)}{\sum_k nk(d)} \tag{3.1}$$

where $ni(d)$ is the number of times the term $i$ appears on document $d$, and $nk(d)$ is the total number of terms in the document.

$$IDF_i = \log \frac{|D|}{df(ti)}, df(ti) = |d : t_i \epsilon d|, \tag{3.2}$$

where $|D|$ is the total number of documents and $df$ is the document frequency of term $i$ (number of documents $i$ appears on).

To calculate the similarity between documents, one can represent them as vectors of tf-idf values where each term is a dimension on the vector space. To find the document more similar to the query, one uses the tf-idf values of the query to create the query vector. The smaller the angular distance between queries and document vectors, the more similar they are.

**BM25:** Probabilistic model that uses TF (term frequency) and IDF (inverse document frequency) functions to rank documents according to the terms from the query. This retrieval function uses a bag-of-words approach, which means the inter-relation that each term have inside the documents is not taken into account. The documents are perceived as unordered collections of terms; there is no idea of a sentence for instance. TF-IDF as BM25 uses this bag-of-words approach. The implementation and assumptions made for this model are described in detail in [57] and also [46].

BM25 is widely used due to its simplicity and effectiveness in the retrieval of documents. The equation used to calculate the similarity between a document (D) and a given query (Q) with BM25 is as follow:

$$Score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)} \tag{3.3}$$

where $f(q_i, D)$ represents the frequency of the term $i$ of query $Q$ on document $D$, $avgdl$ is the average document length on the text collection, $|D|$ is the number of documents on the collection. $k1$ and $b$ are constants with a default value of k1 = 1.2 and b = 0.75. Where parameter $k1$ increase so does TF importance increase for the model. The $b$ parameter is not that trivial but as [45] explained, $b$ helps on reshaping the overall algorithm awareness of document size. Using the default value normally gives more relevance to average sized documents, while with values between [0.3;0.6] is ideal to help level long and average sized documents relevance.

**BM25L:** Due to the Term Frequency (TF) normalization made by BM25, long documents tend to be scored poorly, because a term relevant on a long document will appear less often (as a percentage of the terms in the document) than a relevant term on a small document. Thus, small documents will end up being scored higher than long documents with the same number of relevant terms. To overcome this problem, BM25L [45] was created with an improved TF normalization equation, adding a new constraint to avoid overly-penalizing very long documents. It uses a shift parameter $\delta$>0 to ensure that a

positive lower bound exists on the new TF normalization equation. It makes sure that small values on the TF normalization on BM25 will now be favored, intuitively giving higher score to bigger documents that will normally score very low in terms of TF on BM25. This model uses $b$ values between [0.3;0.6] as said before, to ensure even better results for long documents. This model is an expansion of the BM25 that tries to boost long documents so that these documents can be better interpreted by the retrieval model, increasing the performance of the IR system over BM25 in some cases.

**Language Models:** The Language Based Retrieval Models, as [46] explains, use the context in which each of the terms appear, to classify the similarity between documents, instead of only taking in account the frequency of terms appearing in the collection and documents (as in Bag of words). Language models uses a few primitives from the probabilistic models like the term frequency, but in a different way from TF-IDF and BM25: the probability of a certain term appearing after or before a term or sequence of terms. It is used to mark the document as relevant if the query sequence of terms exists in the exact same order. On this work, two language models will be used: Language model with Jelineck-Mercer smoothing (LMJM), language model with Dirichlet smoothing (LMD).

### 3.2.3 Query Expansion

This is an IR method in which the initial query is reformulated adding new terms related to the query itself. Query expansion can be done by doing global analysis of the collection of documents as described by [67], using dictionaries to add terms related to the query topic (as [5] experimented), natural language synonyms, etc. The primary purpose is to bridge the gap between queries and documents by finding similar concepts that are represented as different words.

### 3.2.4 Rank Fusion

Rank fusion enhances the reliability of the system produced, combining multiple rankings lists produced with many retrieval methods (analyzer, retrieval function, IE algorithms). The goal is to leverage on multiple techniques to increase the overall performance of the system [39].

Rank Fusion can be divided into two groups: score-based fusion (i.e. based on the score provided by the retrieval function) and Rank-based Fusion (i.e. based on the rank provided by the retrieval function). Systems like the one developed by [28] create multiple simulations for comparing multiple rank fusion techniques. A notable result that can be extracted is that when dealing with ranking lists with more than 500 elements, rank fusion algorithms tend to overcome the performance of score fusion algorithms. To corroborate this results [51] and [52] applied multiple types of Score fusion(CombSUM, CombMAX and CombMNZ), and 3 types of rank fusion algorithms (RRF, ISR, RR). In all cases the Rank based algorithms outperformed the score based algorithms.

Reciprocal Rank is the most basic rank-based fusion, in which both RRF and ISR were based. It simply sums the inverse of the document $i$ ranking in every retrieval function used on the rank fusion, as equation 3.4 shows where N(i) represents the total of ranking lists used and r(i) is the document actual ranking.

$$RR(i) = \sum_{k=1}^{N(i)} \frac{1}{r(i)} \tag{3.4}$$

The Reciprocal Rank Fusion [22] is one of the most used algorithms for fusion (equation can be seen on 3.5), which performed well with multiple systems as for example [49, 50]. RRF was a technique applied on both top performance runs on both years. It is very similar to RR the only difference is that a fixed value $k$ is added to the ranking, so that top ranked documents do not get over benefited, normally k=60 for better results as described by [22].

$$RRF(i) = \sum_{k=1}^{N(i)} \frac{1}{k + r(i)}, k = 60 \tag{3.5}$$

The Inverted Squared Rank (ISR) takes the inverse ranking approach of RR and RRF, with the frequency of ranking lists the document appears, so that documents who appear more often on the ranking lists get boosted,

$$ISR(i) = N(i) \sum_{k=1}^{N(i)} \frac{1}{r(i)^2} \tag{3.6}$$

ISR was the algorithm with the best performance on [51]. Rank-based algorithms not always perform better than score-based ones. The Luxembourg Institute of Science and Technology (LIST) TREC CDS 2015 [3] and 2016 [2] submissions that used Score-based algorithm, combSUM outperformed RRF-based runs. CombSUM sums the scores of each retrieval functions for a given documents, as equation 3.7 shows, where $S_k(i)$ represents the score of document i on a given ranking list.

$$CombSUM(i) = \sum_{k=1}^{N(i)} S_k(i) \tag{3.7}$$

In NovaSearch TREC CDS 2015 submission [50] a new algorithm for supervised Rank Fusion was developed, called Learning to fuse (L2F) that used the results of multiple systems fusion to infer which of the fusions are best for the specific problem, in terms of number of retrieval functions to use on the fusion and which retrieval functions to use.

## 3.3 Learning to Rank (LETOR)

Learning to Rank is described by [40], as the field that emerged form the intersection of machine learning applications, information retrieval (IR) and natural language processing (NLP). The purpose of learning to rank is in a wide sense, to use machine learning techniques to perform rankings. The basic idea is to create ranking models to be used for

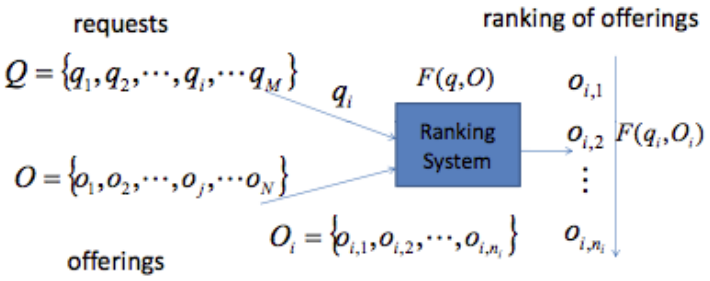


Figure 3.2: Ranking creation sub task. Figure extracted from [40]

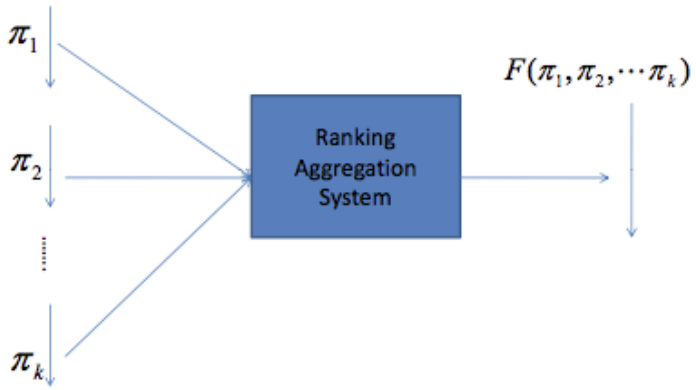


Figure 3.3: Ranking aggregation sub task. Figure extracted from [40]

ranking creation and ranking aggregation. Two different sub-tasks within Learning to rank, detailed on [40].

**Ranking creation:** Consists in the creating rankings of objects (commonly documents) for a given query, based on the information extracted from IR and NLP techniques. This sub task is well described on [40, 42, 43] and is also implemented in both papers. For the purpose of ranking creation, [42] show a quick and easy implementation of a learning to rank system. Figure 3.2 shows how rank creation sub task works. Given a query ($q_i$) from the set Q and a set of offerings O for each query, all offerings ($o_i$) for that query are ranked according to it.

**Ranking Aggregation:** The creation of a global ranking list, given a set of ranking list already computed (independent of the query). More details are available on [40]. The output of ranking creation can be used on ranking aggregation. Figure 3.3 shows how this sub task works, where each $\pi_i$ is a ranking list, regardless of the query. The final ranking list is an aggregation of all previous lists.

Ranking creation is the learning to rank sub-task fitted for this thesis problem, and is the one that will be described in more detail throughout this section.

### 3.3.1 Rank Creation Framework

To elucidate the development and understanding of the rank creation algorithms, it is essential to follow some abstract structure in terms of components needed, and the disposition/interaction of these components. On [43] and [40] that structure is called rank creation framework. Figure 3.4 shows the basic structure of a learning to rank system, and their subsystems interaction. This framework is formed by two systems: Learning system and Ranking system. The first component is the learning system, responsible for the creation of an hypothesis (model) from the hypothesis space. It takes as initial insight a relevant part of the dataset, the training data. There is no fixed size for the training data, but it must represent a meaningful percentage of the queries with their associated list of "offerings" (documents). Too little data results on a biased learning process, producing hypothesis not fitted for a "good" overall prediction of the rank of documents. The second component of the framework is the ranking system, which applies the model previously trained by the learning algorithm, in order to predict the labels/values of the training set. To assure the quality of the models used, a loss function is used to choose the models that best learned the correlation between training set and ground truth. These models are then used on the ranking system to predict the test set documents ranking, based on the testing set insights.

The ranking creation sub-task of learning to rank, which is commonly a supervised learning task, can be implemented by multiple algorithms. But in order to correctly implement it, these methods must always follow two simple proprieties [43]:

**Feature based:** The documents being used ("offered") for the learning process on each query, must be characterized by a vector of features that can be extracted by various applications. This features can be any kind of score, or statistical information that could describe either the occurrence of words on documents or in the query. The features are the learned properties for each entry of the training set, the discrepancies across features values will be understood by the system and learned to predicted results for future queries. Different implementations of learning to rank systems [40, 42, 43, 63] show the wide variety of features that could be used to perform learning to rank, including: Term Frequency (TF), Term Frequency Inverse Document Frequency (TF-IDF), BM25 score [57], Language Models.

**Discriminative training** is the core of the learning to rank system. It models the dependence of target variables $y$ on observed variables $x$ computing the conditional probability distribution $p(y|x)$ to predict $y$ values. This learning process is subdivided in four major components: Input space (documents described by a feature vector), output space (the learning target, e.g. a continuous value, ranked list of documents, target label), hypothesis space (the models created by the learning process), loss function (function responsible for measuring in what degree the model is in concordance with the ground-truth). The idea in learning to rank is to perform

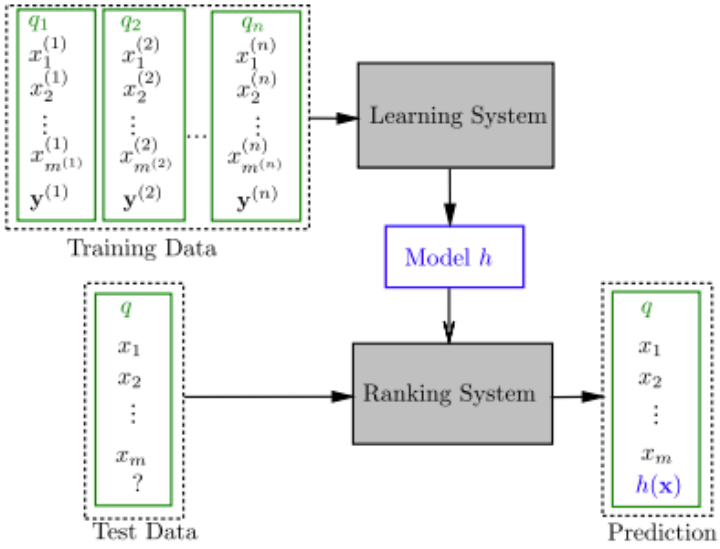


Figure 3.4: The basic structure of a learning to rank system. Originally from [43]

discriminative training with feature vectors from a (document, query) pair as input space, and a ranked list of documents as output space. LETOR algorithms are responsible for creating an hypothesis space, and utilize a loss function to choose the best hypothesis. RankBoost [29], Ranking SVM [17] or LambdaMART[14], are just some of the many algorithms developed for this purpose.

### 3.3.2 Decision Trees Learning

To implement learning to rank frameworks, multiple predictive models can be used. One of the most used models to develop learning algorithms is the decision trees model. The basic concept of a decision tree, as said by [12], is the construction of tree structured rules. The algorithms developed with this structure as backbone, have one goal in common: find a systematic way to create rules/boundaries to nodes in order to also systematically predict classes/values.

The induction tasks performed with Decision trees are a commonly used approach to the wide spectrum of machine learning problems [55]. These structures consist in a tree-like graph where as shown by [55] they consists in a group of nodes interconnected in a exponential manner, each node has 2 child nodes until the final nodes (leaf nodes) are reached. In order to perceive the initial information (training data) and predict results upon new input these trees create on each node thresholds for the input data, an opting for the left or right child according to a specific feature value for a given input element. These Input values (features values) can either be statistical, textual information or even Boolean references, extracted from each element of the dataset, as [43] experimented where multiple values are used as feature. In order to produce for each element either: A label, grouping them according to the labels; Continuous values, enabling the creation of

rankings for the elements. The basic idea of the trees is take the complexity of the job to the nodes, each one performing simple tasks and passing the flow to the following branch (node). Where these tasks are simple conditions checks used to predict results for each input element (e.g. probability of outcome, costs of a given outcome, utility).

Two main types of trees exist, according to the output format of the problem faced: Classification trees predicts the class of a certain point (dataset element), where each classes is a pre-defined label for a subset of the dataset. The nodes of the tree propagate the flow but only on the final node the point is assigned to a class; Regression trees in other hand perform the same calculations but instead of leaving the result to the final node, each node calculates a part of the final result, according to the feature value checked on it. This process results on a continuous value for each point from the dataset. These two types of trees will be used with other machine learning methods called ensemble methods in order to improve the performance and are detailed on 3.3.3

To perform supervised learning on these structures, the first step is to create and train a predictive model [42]. Ground-truth is used to assess how feature values are distributed on the training set (subset of the dataset), training the model with the correlation between features and the element's ground-truth description (e.g. class, continuous value). Only after the creation of the model (training step), the decision trees can be used to predict the results of each individual entry of the dataset. On the LETOR context seen on [43] and [42] these entries are documents, and the ground-truth is a document containing each document relevance to a given query.

When testing, the feature values are processed by nodes throughout the tree model that had already learned the training set, and according to the information extracted from the training set the model will either classify the element or give it a single continuous value.

### 3.3.3 LETOR algorithms approaches

Learning to rank algorithms are based on general machine learning algorithms as for example: Decision trees, Support vector machine, Naive Bayes classifier. As said on the previous sub-section 3.3.2, the algorithms used on this project are all based on decision trees predictive model. However, regardless of the type of model used, all algorithms fit into three specific approaches: Pointwise, Pairwise and Listwise approaches.

**Pointwise approach** This is the more straightforward of the approaches. It applies commonly used learning algorithms to perform classification, regression and ordinal classification on each point (document) of the dataset, predicting for each document its relevance to the query. Since this approach doesn't take into account the group structure, the loss function used to select the best hypothesis at learning time must be applied to each object individually. Consequently the best models will be selected according to the results on each of point presented on the training set.

**Pairwise approach** Instead of assessing the relevance of each individual object (documents), the pairwise approach checks the relative order of a pair of documents, compared with the ground truth relevance. Assuming there's a document $di$ more relevant than document $dj$, the pairwise approaches loss functions and scoring functions will discriminate the hypothesis, according to the hypothesis relevance assumptions made for documents $di$ and $dj$. This is a more natural approach to the learning to rank problem, when compared with the pointwise approach, being able to design hypothesis based on the documents relations and was the natural next step after pointwise. Due to that, this approach is one of the more used for development of algorithms: some of the more known algorithms are build on this pair schema. For example: RankNet [13], RankBoost [29], LambdaMART [14]. LambdaMART was the best algorithm for the "Yahoo! learning to rank challenge" [19].

**Listwise approach** This approach is the most natural when facing a ranking problem: instead of using single documents or pairs of documents, the training and prediction instances are ranked lists of documents. To perform learning on the training set, each feature value (document) gets a score, and later, they are ranked according to this score. This has been the most developed class of algorithms in most recent years. For example: AdaRank [68] and ListNet [18].

Both pointwise and pairwise approaches solve problems by performing multi-class classification, regressions and ordinal classifications. Solving ranking problems as regression can be done by assigning to each object a real number, according to its relevance to the query (for example: 0, 1, 2). Subset Ranking with Regression developed by [23], and Polynomial Regression Function by [31] are two examples of algorithms that utilize the regression approach for learning to rank. When using Multi-class classification, the idea is to classify each instance as one of $n$ classes for the problem, and later sorting elements inside each class by using equation 3.8.

$$f(x_j) = \sum_{y=0}^{y-1} y \cdot P(\hat{y} = k) \tag{3.8}$$

where $x_j$ represents a document from an instance, $\hat{y}$ is the prediction class of that document and k is the number of classes existing, the higher the probability of a documents belonging to a class, higher the rank inside that class. Knowing that each class will represent levels of relevance, for example from 1 to 5 where 5 is high relevance and 1 is no relevance. The last one, ordinal classification consists on assigning to each instance a continuous value, that must be representative of the relevance of the instance to the query. To achieve the final ranking, all this values most be sorted by this value.

Listwise algorithms, which address instances as ranking lists, can't directly apply classification or regression solutions for machine learning problems, because they do not intend to give a class or a value to each element individually, since it uses ranking lists as elements. This type of algorithms follows two different ideas, as [43] described:

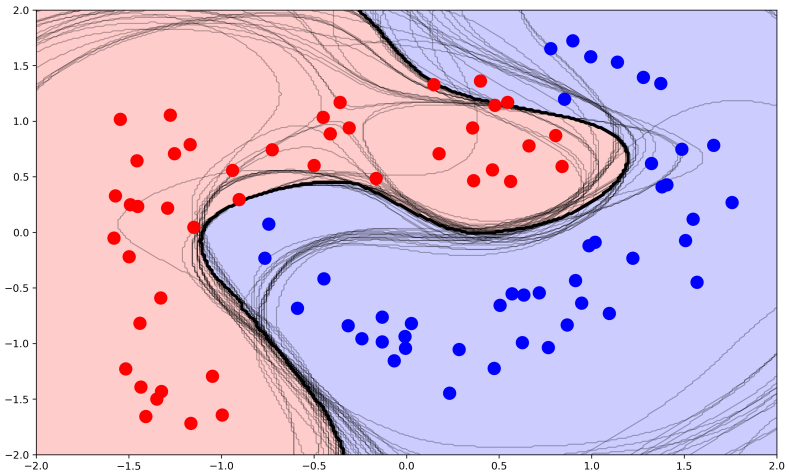


Figure 3.5: Example of bootstrap aggregation used for classification

Optimization of IR Evaluation Measures and Minimization of Listwise Ranking Losses. The first as the name implies, tries to optimize information retrieval metrics as NDCG or Mean Average Precision (MAP) to perceive which ranking list were better according to IR metrics. The other approach goes by the same ideology but instead uses a lost function to measure inconsistencies between the ranking list and the ground truth.

### 3.3.4 Ensemble Methods

On regular machine learning algorithms, given a training set, a classifier (hypothesis) is created for predicting each input instance label for an unseen query. Ensemble methods try to optimize the accuracy of the final hypothesis by creating and converging multiple hypothesis into a single one, in multiple possible ways. The goal is to improve effectiveness of the hypothesis created by algorithms without ensemble methods, by applying this methods to hypothesis created by regular ML algorithms. Ensemble techniques are used across nearly every machine learning sub task (learning to rank included). To achieve success on the optimization of classifiers predictions, as [26] explained, these methods are restricted by a couple of conditions, to either maintain or improve quality of the final hypothesis. First, all individual classifiers created must be accurate (i.e, maintaining the error rate below 0.5). This ensures that the final classifiers is a merge of accurate classifiers. Second, all classifiers must be diverse. So, different classifiers must produce a wide variety of errors on the same input data. This ensures that errors are not propagated.

The following are some of the more commonly used ensemble methods, as detailed on [26]:

**Bayesian Voting: Enumerating the Hypotheses** Is a set of statistical methods, in which as all hypothesis $h$ are defined by: $h(x) = P(f(x) = y|x, h)$, where x is the input data, h is the hypothesis and y is the prediction. Tuning this to the problem of ensemble all this hypothesis can be seen as computing $P(f(x) = y|S, x)$, where S is a new

training set sample (system). This ensemble consists on the sum of all hypothesis weighted by their posterior probability $P(h|S)$, represented by: $P(f(x) = y|S, x) = \sum_{h \epsilon H} h(x)P(h|S)$.

**Manipulating the Training Examples** This method consists on creating multiple different subsets of the same training set, and converging the hypothesis, created by each of the new subsets, into one final hypothesis. Algorithms like bootstrap aggregation [10] or adaptive boosting [30], are some of the most used algorithms using this technique. Each algorithm creates training subsets in his own manner and using them in different ways. Figure 3.5 is a good example of bootstrap aggregation applied to classification. Where we have two classes (red and blue) of points, and each line represents a classifier (hypothesis). The final hypothesis is a merge of all those lines splitting the two classes of points.

**Manipulating the Input Features** This feature selection technique consist on changing the feature vector, which is used to detail each of the entries of the dataset. This method implies that not all of the features used are truly relevant for the model creation. As this redundant features can cause under-performance of the learning algorithms, the idea is to remove features that are not relevant for the problem. The new subset of features is expected to benefit the creation of the converging hypothesis. If a better hypothesis cannot be achieved with a new subset of features, then the complete feature values set is used.

**Manipulating the Output Targets** Based on the number of classes $K$ for the problem, it creates for example two subsets were the classes are randomly divided based on the labeled classes. For example, inputs labeled as k=1;2;3 are all in one subset and k=4;5;6 in the other subset. This subsets are the only classes presented to the learning system; the ensemble part happens by converging all the hypothesis created, each one with two new randomly created subsets of labeled data.

**Injecting Randomness** Consists in inserting any type of randomness to the learning model. It can be made by adding previously predefined values in a random fashion, like initial weights for an algorithm or randomly selecting the features that will be used on the learning process. While being widely used for its simplicity, this is not the best ensemble method, since most of the previous used methods test the quality of the new random value, while this method just simply inserts randomness.

## 3.4 Clinical Information Systems

A lot of work has been done on the field of information retrieval for medical literature in the last decade, as [27, 58, 59] all systems and system components to be used for biomedical information retrieval, focusing on very different IR system components. These components can help the retrieval of biomedical documents, either with new algorithms

and methods developed to improve a wide range of IR systems, or development of topic related implementation, creating more tailored IR systems.

### 3.4.1 Clinical Information Extraction

To retrieve query related topics, it is indispensable to have a good information extraction implementation, as retrieval is only as good as the information it can be extracted from the documents. Information Extraction (IE) processes unstructured text from collection of unstructured data and semi-structured data. When dealing with unstructured data, a full-text processing will be made to extract the relevant information from the document. This is a hard problem, as the system has no prior idea of what is discussed on the document, and means that a lot of information can be incorrectly classified or discarded. A good example is the RobotReviewer framework [47] detailed on section 3.6. In RobotReviewer, references of Population, Intervention, Control and Outcome (PICO) of Randomized controlled trials (RCTs) are extracted without any feedback from the user, using machine learning and NLP to do a full-text processing and search. Not all RCT get good extraction results, as there is no way of knowing if all the PICO fields are explicitly mentioned on the document.

When dealing with semi-structured data (i.e. documents that have separated fields some with narrative note, written text, or specific fields like gender: Male or Female), one knows which fields characterize the document. This means that a lot of new retrieval ideas can be used, enabling the creation of systems such as the one created by [36] where a real-time monitoring system is created using the information stored on the databases of Hospitals Intensive Care Units (ICUs). Using information extraction similar to the one used on Apache cTakes, machine learning regression and classification to fulfill the ICU needs. For example, using classifications to understand if the patients respiratory health and heart rate is inside life threatening values or not. The important part when dealing with this type of collections is to study the dataset and understand what is relevant for the problem.

As with most of TREC tracks, the Clinical Decision Support track (CDS) and Precision Medicine track (PM) datasets are composed by a collection of semi-structured documents, containing narratives notes (e.g. full text of the purpose of the study, summary of the intervention, outcomes) and fields characterizing the article (e.g. MeSH terms, gender of patients, age). Both CDS and PM tracks are composed of biomedical articles from the Open Access Subset of PubMed Central (PMC). The PM track is subdivided into two tracks: one to retrieve biomedical articles and other to retrieve clinical trials from clinicaltrials.gov database. The main difference between both tracks is that the CDS is focused on answering general medical questions, normally concerned with diseases that have a consensus of multiple acceptable interventions to treat patients. On the other hand, PM as the name suggest, tries to answer very specific questions, retrieving information about oncology diseases. Information that need special attention due to its severity and the lack of proven good interventions. That is why the clinical trials are very important to TREC PM, where

new trials can unveil new treatments.

The papers [49, 50] are 2014 and 2015 submissions of NovaSearch, group from Universidade Nova de Lisboa, for TREC CDS. Both papers show how documents fields can be used to extract data, that will be used on the text-search engine and for search time filtering of documents. For example, on the 2015 submission, the fields that describe the document itself like summary, title and description will be used on the search, while fields that give the information about the journals where the articles were published, were used to create a search-time filtering called Journal-based filtering. This excluded articles from journals deemed irrelevant for the specific query type. Excluding criteria is a type of filtering that can be applied to the results. Even if some similarity is found between a document and the query, the exclusion criteria extracted from the document can exclude it automatically. Filtering the results is one of the basic techniques that can be applied to information retrieval to guarantee a minimum limit bound on efficiency of the IR system. Belkin [7] study shows that a system without filtering is more likely to perform worse than one with filtering.

This type of implementation can be used in a lot of different approaches, leading to performance boosts on information extraction. But dealing with the given fields and making no assumptions of their interconnection leaves a lot of relevant information hidden in both unstructured and structured fields. This means that feature extraction ideas could be applied taking a combination of fields to infer new features. The top ranked-team of TREC Precision Medicine track, from the University of Delaware [1], used a different approach where besides storing the search fields and those that will later be used for filtering at search time, they grouped four fields: Primary Purpose, Intervention, Study Type and Phase (of treatment). What they discovered is that when these fields are within a range of specific values, the trials normally are relevant to oncology treatments. Their algorithm gave trials that contain them an increase in score, which will be added to the scoring of the searching fields. The following are the values that grouped all together indicate a oncology treatment:

- **Study type**: Is the type of medical study performed on the clinical trial. For the purpose detailed before, the trials with study type: intervention, are given an extra boost;

- **Intervention**: A medical intervention is the act of curing or helping patients with direct actions to ones condition. This can be done in numerous ways, but when we are dealing with cancer patients few options remain. Only drug, biological or radiation treatments are performed, so clinical trials with this type of interventions have higher chance of being cancer related;

- **Phase**: The phase of the clinical trial. As shown on figure 2.2, 4 types of clinical trials exist. Patients are assigned to phases according to their condition and the study itself. Cancer related trials are normally around phases 2, 3, or 4.

- **Primary purpose**: The principal idea behind the clinical trial. When dealing with cancer related treatments the primary purpose is treatment.

One final and more complex approach that can be used either with unstructured or semi-structured collections are the Part-of-speech taggers such as the Stanford POS tagger [62], or other frameworks like Apache cTakes [58]. They process the information and extract the relevant entities referred on the texts. These techniques benefit from using Unified Medical Language System (UMLS)-like dictionaries, to store unambiguous UMLS ids for the given disease instead of the retrieved terms. So the more specific the dictionary better the results. In [37] and [44], Human disease ontologies are created or improved based on multiple biomedical resources, each one with different approaches. Using POS taggers enables matching documents that describe similar diseases and procedures using different terms, creating better links between query and collection documents for searching and indexing with automatic biomedical term extraction.

### 3.4.2 Clinical Query Expansion

The key idea behind query expansion is to fill the gap between the terms in the query and those on the collection, by adding terms related to those on the original query in order to get a wider range of results. For example a query for "Car" can also benefit by querying the system for "Car" and "Automobile" simultaneously. To perform this technique, two major mechanisms can be applied: Analysis of terms on the collection and Domain specific Ontologies.

Analyzing terms on the collection to perform query expansion can be done using local and global analysis. As described in [67], Local Analysis combines the query with local feedback results (Documents chosen by the user), extracting the top terms of the selected documents. One very popular method is used to provide users with automatic local analysis, pseudo-relevance feedback (PRF). It consists in adding to the query the top scoring terms of the top retrieved documents, using the initial query and not chosen by the user directly. This method was used on [52], NovaSearch CDS 2015 and 2014 [49, 50] with an Apache Lucene method designed for PRF, called more like this (MLT), that provided users with a parameterizable method to perform PRF query expansion. Besides these three previous papers, [69] and [53], also submissions for TREC CDS 2014 track, used PRF and the results show that the system performance benefits from it. If PRF is not carefully parameterized, it can have the opposite result, downgrading the performance of the system as shown on [16], by adding new terms that may be irrelevant to the initial query. On other hand, Global analysis processes the corpus of the collection to infer which terms should be used on the new query, perceiving the co-occurrence of query terms and these new terms around the collection.

Local context analysis, based on both previous analysis, selects expansion terms based on co-occurrence with the query terms (global analysis) within the top-ranked documents (local analysis). It as performed better than any of the other analysis as shown in [67], where

local context analysis improved the performance of the query expansion in comparison with any of the previous techniques.

An alternative to local and global analysis is to use the benefits of Language Models to use the context of the terms to achieve better results on the Ranking functions, instead of simply applying a pseudo-relevance feedback to perform query expansion with local and global analysis. The authors in [6] proposed algorithms for Query Expansion that were aware of the interrelations of the words and based the system on the co-occurrence of terms on the same context. The expansion could be either with a creation of a HAL (Hyperspace Analogue to Language) space, where words were perceived as a vector of co-occurring terms and their respective weight. Another expansion method uses what is called information flow that creates queries that can be seen as part of the same information flow, so all terms must be correlated according to the corpus and the furthest the terms on the new query, the more sparse their co-occurrence is. This algorithms shows what advantages can be taken from using language models on Query expansion and how to use them.

Not all expanded query are assumed to be good. Based on this idea, [24] developed an algorithm that tries to ensure that IR system gets the best possible results using QE, even if this means to not use the expanded query. Cronen showed a method of inspecting the results of both unexpanded and expanded query and infer which of the ranking lists best answer the initial query.

All of the previous methods used to perform query expansion can improve the retrieval performance but are based on information present in the documents on the Corpus. For expert domains such as the biomedical domain, a lot more can be done with external sources of information to help create better IR systems. External biomedical ontologies like SNOMed CT and MeSH are commonly used to fill the semantic gap between query specific terms (terms that could be mentioned on those dictionaries) and their alternative, preferential (label that more correctly identifies the given term) or synonyms terms. These terms are the ones that will be used to perform query expansion, adding them to the initial query. A considerable amount of work have been done on this field. For example in [52] the query was expanded with only synonym terms from MeSH. While on [20] only the MeSH MajorTopic terms were used, both on [20] and [52] the results show that the performance of the system benefits from the usage of MeSH terms. [49] used the preferred and alternative terms, while [50] used all the preferred, alternative and synonyms provided in MeSH and also used terms and relations of the Web Ontology Language (OWL) version of SNOMed CT, achieving good results on the competition.

Other approach to use these external sources may be seen on [1] where these dictionaries are always used in all system to give more specific terms for the Disease expansion and not for the overall query. This distinction was made to ensure that only topic related terms are expanded, since the disease field description is not ambiguous.

## 3.5 Evaluation Test-beds

This section will detail some of the more relevant tracks for which the previous explained systems were developed. The goal is to give insights on the Datasets and how they were structured, the primary purpose of each challenge and the test queries used for submission.

### 3.5.1 Medical ImageCLEF 2013

The goal of the medical retrieval task of ImageCLEF 2013 is to find images and textual information from the subset of PubMed Central containing 305,000 images and journal articles [34]. For ImageCLEF 2013 there were 4 subtopics (Modality Classification, Compound figure separation, Ad-hoc image-based retrieval and Case-based retrieval) but the only relevant for this work is the Case-based retrieval, as it is also focused to the textual retrieval. This task contains 35 topics (case description) with patient demographics, limited symptoms and test results including imaging studies, but not the diagnosis (they will be extracted from the retrieved cases/journal articles).

A query for the Medical ImageCLEF 2013 is for example figure 3.6 accompanied by a text with description: "A 43-year-old woman with painless, gross hematuria..."

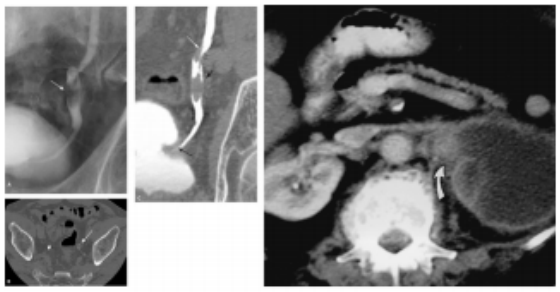

Figure 3.6: Medical ImageCLEF query example.

### 3.5.2 TREC CDS

The Clinical Decision Support (CDS) Track is focused on the retrieval of biomedical articles for answering generic clinical questions about medial records. The dataset for the Task is a snapshot from January 2014 of the Open Access Subset of PubMed Central. Both 2014 and 2015 use the same subset with 733,138 full-text biomedical articles. Topics are medical cases narratives containing two fields description and summary of the case. Both are divided into multiple types of topics: 10 Test topics, 10 treatment topics and 10 diagnosis topics. Each topic represent the purpose of the physician on the case, describing either if the physician is looking for tests to experiment with the patient, treatment to perform on the patient or simply the diagnosis of a certain case. The Track is subdivided in two tasks A (no diagnosis field on the test and treatment queries) and B (explicit diagnosis field on the test and treatment queries), the first one is the only task for CDS 2014 and CDS 2015 contains both. Queries with test and treatment have a diagnosis annexed to it, which

may improve the overall performance of the retrieval system, giving more precision to the query. The following is an example of a test topic for TREC CDS 2015:

Listing 3.1: Example of query provided in a xml document showing the topics structure

```
<topic number="19" type="test">
<description>
A 66yo female with significant smoking history...
</description>
<summary>
A 66-year-old female smoker presents with worsening dyspnea...
</summary>
<diagnosis>Chronic Obstructive Pulmonary Disease</diagnosis>
</topic>
```

### 3.5.3 TREC PM

This Track aims to provide physicians with a system capable of retrieving article and trials for very specific use cases, Oncology cases. It contains 2 datasets each one for the 2 tasks that make part of PM 2017:

- **Scientific Abstracts**: Articles from the biomedical domain, targeted toward cancer therapy. January 2017 snapshot of PubMed abstracts, which additionally contains abstracts obtained from AACR and ASCO (Associations totally dedicated to cancer research).

- **Clinical Trials**: Documents detailing observations and experiments performed on patients to answer specific questions. April 2017 snapshot of from *ClinicalTrials.gov*.

The topics contain four fields: Disease, Gene Variant, Demographic and Other. Disease is the actual cancer the patient suffers from (e.g. Melanoma, Liposarcoma), the Gene Variant is the actual gene which led to the cancer, Demographic is age and gender specifications of the patient. The Other field contains key exclusion criteria such as other diseases or important information of the patient record that may change the relevance of some documents for this specific patient. The corpus and queries, detailed on figure A.4, that were used on this work was the TREC PM 2017 clinical trials dataset. The final number of queries was 28; queries 10 and 12 were excluded from testing since no relevant documents could be found on the collection.

## 3.6 Clinical Information Processing Frameworks

Multiple systems have been designed to simplify the applications of these techniques. The detailed frameworks are analyses and knowledge systems used to perform information retrieval.

### 3.6.1 Apache Lucene

Lucene is a library for information retrieval that can provide users with a full-text search, with High-Performance and scalable indexing tools and search techniques which could be matched to the requirements of the system. It supports most of the previously described indexing and search methods (e.g. inverted index, retrieval methods). It can work with multiple types of data, enabling the creation of numerical range queries, specific textual queries (e.g. is gender equal to "male" or "female") and textual queries. This framework uses 4 types of methods to perform textual queries:

- **Should:** Scoring documents according to their relevance to the query, but not decreasing the score or excluding documents with low relevance.
- **Must Not:** Excludes documents that contain query terms. Used only as a filter, not changing the score of documents at all.
- **Must:** Increase the score of documents which contain the query terms, and downgrades the score of documents not showing query terms.
- **Filter:** Excludes documents with no relevance to the query. Doesn't change the score of documents.

### 3.6.2 RobotReviewer

Evidence Based Medicine (EBM) as the names suggests, is the approach to medical practice that uses evidences from cases and scientific research to help the physicians in the decision making. For that purpose, systematic reviews are commonly used EBM tools: they critically analyze literature in multiple manners providing an exhaustive and complete summary of it.

RobotReviewer [47] aims to semi-automate the systematic Randomized Controlled Trials (RCT) review creation process, extracting from the trials multiple information and synthesizing it in two ways:

- **Quantitative**: PICO (Population, Intervention, Control, Outcome) process aims at getting a sense of what exactly as been wrote on the clinical trial. A good example is the work developed by [59], where PICO annotations were extracted and used to frame and answer clinical questions.
- **Qualitative**: To perform systematic reviews, it is very important to take notice of the correct formation of the RCT's. Badly created reviews may be biased, which may lead to distorted results (e.g. if the patients are aware of the type of study and the types of treatments being made, placebo effect could occur). For that, RobotReviewer also extract four bias fields to ensure the quality of the study: Random sequence generation, Allocation concealment, Blinding of outcome assessment and Blinding of participants and personnel.

Robotreviewer utilizes machine learning techniques and combines them with NLP to extract the information from the RCTs in a simple way, with no technological knowledge needed to use the software.

### 3.6.3 Apache cTakes

This software was developed by [58], to extract the information from multiple types of clinical records. It processes the clinical notes using Apache OpenNLP technology and Machine Learning techniques. Even though it is based on similar tools as RobotReviewer, the goal is different. Apache cTakes uses this tools to identify clinical named entities like diseases, symptoms, drugs, dosage, procedures and much more entities relevant in a clinical point of view.
cTakes is made up by multiple components each extracting different insights from the clinical note, the following are examples of these components:

- Drug Named Entity Recognition
- Part-Of-Speech Tagger
- Context Dependent Tokenizer
- Others

This software can be used to compute multiple types of medical related dictionaries. One good example are ICD-10 dictionaries, that gives codes to multiple medical entities (e.g. diseases, drugs, symptoms, social circumstances) and facilitates the retrieval process on the literature that uses this types of dictionaries. For example, if the patient as a condition on the respiratory system a code between J00–J99.

## 3.7 Critical Summary

This chapter showed some of the more commonly used techniques for information retrieval and for the study fields surrounding it. What is very much noticeable on information extraction is that most of the systems try to overcome their problems by using as much structured information as they can. This means that fields that are extracted are either to perform a full-text search on a specific field, or to match information from specific fields (e.g. MeSH terms, age, condition). This type of information extraction can only lead to results with median precision, as the most explicit information will be on the text fields. On the other hand, systems that deal with full-text search in order to extract characteristics from the text, as described by [47] with PICO annotations, can easily deal with full text characterization. Being nearly impossible to understand human like discrepancies on text, as for example: "the purpose of this trial is strictly for lung neoplasm, not for female breast cancer". Imagine this text is on the exclusion criteria field of an article, using either one of the text searches described for searching "lung cancer" on the exclusion criteria, will

produce a lot of incorrect classifications for the document. So what is needed is a way of processing each line of the given text and infer which words have a certain meaning or have a certain form (positive/negative). Tasks like NER can be very helpful to overcome this problem.

In terms of query expansion, various ideas are present in the literature, from performing collection analyzes to add terms to the query or even using external sources, providing the query with more precise or general terms depending on the IR purpose. PRF is a very interesting technique to perform collection analyzes that was used during the implementation of this work to provide more precise queries. Using MeSH and SNOMED CT terms can also be a very good idea, since the systems normally benefit from it.

Rank fusion have been widely used across the IR community, providing consistent performance improvements when compared to individual system results. Algorithms such as RRF and CombSum are two of the more commonly used, and it will be interesting to measure how these algorithms can improve the global system performance. The difficulty in rank fusion is the same as query expansion. How do we know which of the used methods improve the system? This question can be answered again with a supervised learning algorithm (e.g. L2F).

In terms of learning to rank, given the variety of different approaches within decision trees framework it would be very interesting to understand how each algorithm deal with the problem in hands. Testing how pointwise, pairwise and listwise approaches behave on the thesis dataset. Besides that, the small amount of queries makes this system very good to evaluate and understand how these algorithms deal with small training information, and how can ensemble methods be used to overcome this problems.

CHAPTER 4

# UNSUPERVISED RETRIEVAL SYSTEMS

Throughout the previous chapter, the state of the art of medical retrieval systems was discussed, including some of the more common problems of those systems. The set of problems include: filtering according to demography, using correct medical relevance assumptions and filtering according to exclusion criteria. This thesis goal is to create a system capable of achieving very precise results for the problem of matching patients to clinical trials. The developed systems were based on the TREC Precision Medicine 2017 Track. Table 4.1 shows the notations used throughout this chapter.

Table 4.1: Notation used in this thesis.

| Notation | Definition |
|---|---|
| *ct* | Clinical Trial |
| *Ru* | A set of methods used to create a rank list with unsupervised learning |
| *Ru_methods* | Retrieval run. Methods detail the set of methods the implementation is focussed on. |
| $ct\_field$ | *Field* content given Clinical Trial *ct*. |
| $tp\_field$ | Topic (Query) *field* content. |

## 4.1 Document parser

The documents parser uses a pipeline of filters to mitigate factors like the number of non relevant words and variation of words (e.g. reduce words to their root forms *cancers* and *cancerous* to *cancer*), which leads to more effective retrieval results. The filters used on the benchmarks were as described on Table 4.2. The combination of these filters are

the components of the text information analyzer described on section 3.2.1. As Table 4.2 shows, the filters used have two main principles: Normalization of words and removal of unnecessary words. Tokenization and stemming are example of word normalization. The first is responsible for removing the punctuation and reducing all words to lower case, while stemming is responsible for reducing all words to their root forms. The remaining filters are used to remove stop words, very common words across English literature (e.g. "and", "that", "for"), and convert all text into lower case.

Table 4.2: All the filters used on the Lucene text analyzer.

| **Method** | **Definition** |
|---|---|
| Tokeinzation filter | Removal of punctuation and splits of text by white spaces into chunks. |
| Lower case filter | Converts all text to lower case. |
| Stop Word Removal | Curated list of common words that should be excluded during processing (e.g "this","is","a"). |
| Stemming | English language stemming, used to reduce English terms to their root form. |

## 4.2 Information Extraction and Indexing

To index the information contained in the documents, it was necessary to extract the *ct_fields* that could be relevant in a medical environment. This choice was supported by the Information Extraction examples used on the systems described on the articles on section 3.4.1, and also inspired by some of the oncology principles detailed on chapter 2. The chosen fields to be extracted from each *ct* are detailed on table 4.3.

The format of the documents provided in the collection (a field-structured document), helped us minimize the workload on the pre-processing of the text before indexing it. For example, **<minimum_age>** *18* Years **</minimum_age>**, there is only need to split the text by white spaces and the result is an array containing ["18", "years"]. The specification ensures that the first value on the array is a string containing a number, that represents the minimum age of the patients for this clinical trial.

The information contained in the larger text fields such as, *ct_brief_title* or *ct_summary* is first processed using the analysis process explained on 4.1, and then indexed. Note that not all the fields are available in all clinical trials.

## 4.3 Matching: Retrieval and Filtering

The created runs reflect the impact of multiple methods for information extraction on the retrieval. Table 4.4 details each one of the *tp_fields* extracted from the patients clinical record, used to scrutinize documents in the most relevant manners. The following sections will show in detail the iterative process of choosing the best methods for retrieval

Table 4.3: Detailed description of the fields extracted from the clinical trials.

| **Fields** | **Id** | **Description** |
|---|---|---|
| Official Title | *ct_offTitle* | The official title of the clinical trial. |
| Brief Title | *ct_briefTitle* | A brief title containing only some keywords of the title. |
| Brief Summary | *ct_summary* | A compact summary of the clinical trial. |
| Detailed Description | *ct_detailedD* | A detailed description specifying the goal of the clinical trial. |
| Text | *ct_text* | Contains a concatenation of the text fields: Official Title, Brief Title, Brief Summary, Detailed Description |
| Inclusion Criteria | *ct_inclusion* | Inclusion criteria for the patients (other diseases, allergies, previously used drugs, cancer gene mutations). |
| Exclusion Criteria | *ct_exclusion* | Exclusion criteria for the patients (other diseases, allergies, previously used drugs, cancer gene mutations). |
| Max Age | *ct_maxage* | The maximum patient age on this trial. |
| Min Age | *ct_minage* | The minimum patient age on this trial. |
| Gender | *ct_gender* | The patient gender accepted on the trial. Given the value "1" to female only trials, "2" for male only and value "0" to trials with no gender restriction. |
| Study Type | *ct_studytype* | The type of the study the clinical trial is focused on. |
| Intervention | *ct_intervention* | Explicit the intervention made on the patient (e.g. drugs, radiation). |
| Primary Purpose | *ct_purpose* | The purpose of the trial: the goal is either the treatment, diagnosis, prognostic or prevention of a disease. |
| Condition | *ct_condition* | Keywords for the disease the trial intendeds to treat |
| Condition MeSH | *ct_meshCondition* | Automatic keywords, from MeSH, for the disease the trial intendeds to treat |

An exhaustive benchmarking of such techniques is available on Chapter 6. Regarding retrieval functions, all techniques will be tested in combination with one or more of the following retrieval models: BM25, TFIDF, BM25L or LMD similarity, as they achieved

Table 4.4: Detailed description of the fields extracted from the patients clinical record.

| **Fields** | **Id** | **Description** |
|---|---|---|
| Disease | *tp_disease* | The disease where the patient treatment will be focussed on. |
| Gene | *tp_gene* | Gene mutation associated with the previous described disease. |
| Demographic | *tp_demo* | Age and gender information of the patient. |
| Age | *tp_age* | Age of the patient, extracted from the field *tp_demo* |
| Gender | *tp_gender* | Gender of the patient, extracted from the field *tp_demo*. Given the value "1" to female patients and "2" for male. |
| Other | *tp_other* | Other conditions the patient may suffer, that are not oncology related diseases. |

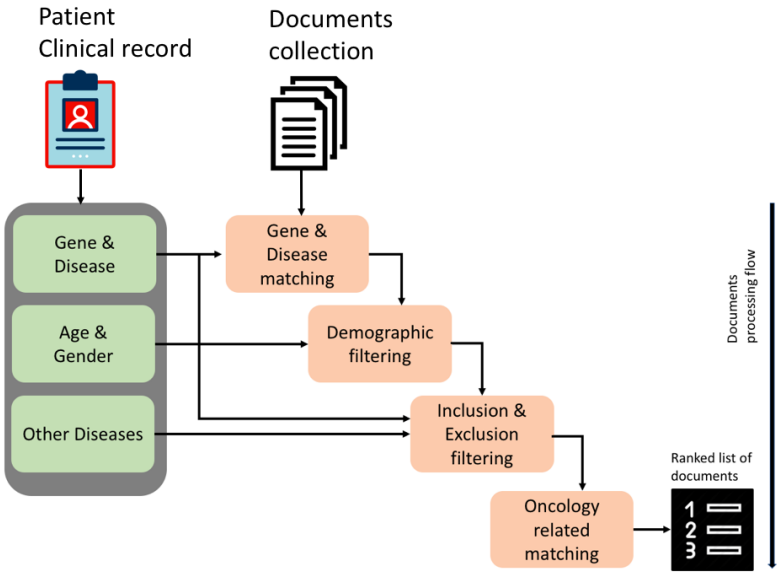


Figure 4.1: Information retrieval system with some of the used features.

good retrieval performance on previous biomedical information retrieval systems [49, 50, 53, 69]. Throughout this section the implementation of each module described on Figure 4.1 is going to be explained, and other documents processing methods that were not present on it.

Table 4.5: Retrieval runs with methods focused on gene and disease search.

Lines in x(n) format, n is the Lucene boost applied to that specific Gene pr Disease match (only present in cases were boost is different than 1).

| Run | Gene | Disease |
|---|---|---|
| *Ru_gen* | x | |
| *Ru_dis* | | x |
| *Ru_gendis* | x | x |
| *Ru_gendisB* | x (2.0) | x (1.5) |

### 4.3.1 Gene and Disease

According to chapter 2 description of cancer diseases and genes, it becomes clear that both are indispensable for finding matches between patients and clinical trials. A trial that neither mentions the patient disease or gene mutation, will probably not be a relevant document for that patient. The initial step for all retrieval methods, is to use both $tp\_gene$ and $tp\_disease$ as query and search for these terms on $ct\_text$. This query was implemented following Equation 4.1

$$Q_i = (tp\_disease \cup tp\_gene; Search\ on: ct\_text) \tag{4.1}$$

Table 4.5 shows the implementations focused solely on gene and disease search, with two *Ru* searching for both gene and disease. *Ru_gendisB* was manually tuned to achieve the best possible results, giving a boost to both gene and disease search.

### 4.3.2 Demographic filtering

Throughout the top retrieved trials from either *Ru* on table 4.5, age and gender stand out as a relevant feature that should be used to improve the retrieval system. Many of the trials incorrectly retrieved as relevant, have age and gender restrictions outside of the desired values. Even if the patient is eligible for a specific trial, age and gender can be outside the intended demographic group.

An inspection of relevant documents (according to the provided relevance judgments on TREC PM 2017 dataset), confirmed that demographic is mandatory for a trial to be relevant. This was the case on all patients on all trials except 10 patient-trial pairs that were, most likely, incorrectly classified as "relevant" (the trials clearly specified a different age or gender from those patients).

All three *Ru* on Table 4.6 apply demographic filtering to the best retrieval function seen on sub section 4.3.1, using $ct\_minge$, $ct\_maxage$ and $ct\_gender$ to check patients eligibility for each trial. Following the Equation 4.2 to filter documents, where only documents with $tp\_age$ and $tp\_gender$ within acceptable values are not filtered.

$$Q_i = (ct\_minage <= tp\_age <= ct\_maxage) \cap (tp\_gender = ct\_gender \cup ct\_gender = 0) \tag{4.2}$$

Table 4.6: Retrieval runs with methods focused on demographic filtering.

| Run | Gene / Disease | Age | Gender |
|---|---|---|---|
| *Ru_age* | x | x | |
| *Ru_gender* | x | | x |
| *Ru_demo* | x | x | x |

Table 4.7: Retrieval Runs with methods, focused on Exclusion and Inclusion criteria.

| Run | Gene / Disease | Demo | Inclusion | Exclusion |
|---|---|---|---|---|
| *Ru_inclusion* | x | x | x | |
| *Ru_exclusion* | x | x | | x |
| *Ru_criteria* | x | x | x | x |

### 4.3.3 Inclusion and Exclusion criteria

One of the most evident flaws on the retrieved trials was the lack of compliance with both *ct_inclusion* and *ct_exclusion* fields. Following the patterns observed on trial data, it is clear that "Exclusion criteria" should work as a filter: if any match between patient and the trial exclusion criteria terms exists, then the trial is automatically non relevant. The exclusion criteria query was built using Equation 4.3, where both $tp\_gene$ and $tp\_other$ are used to search for gene and other diseases the patient suffer from on $ct\_exclusion$, which is expected to not occur. So documents which have a reply for this query are automatically excluded.

$$Q_i = !(tp\_gene \cup tp\_other;\ Search\ on : ct\_exclusion) \tag{4.3}$$

On the other hand, matching "Inclusion criteria" should not greatly impact the direct removal of false positives from the top retrieved functions, as non relevant documents can have inclusion criteria that match the patient condition. Even so, matching terms in the inclusion criteria to patient information is key to corroborate and boost the scores of true positives. Inclusion criteria was implemented following Equation 4.4, where $tp\_gene$ and $tp\_disease$ are used for search references of gene and disease on $ct\_inclusion$. Table 4.7 shows the *Ru* created using inclusion and exclusion criteria filter.

$$Q_i = (tp\_gene \cup tp\_disease;\ Search\ on : ct\_inclusion) \tag{4.4}$$

### 4.3.4 Domain Specific Methods

The core idea behind domain related methods is to retrieve all documents related to cancer treatments, even if the trial is not an exact fit for that specific patient. Since the implementations that are detailed in this chapter are incremental, most top ranked documents are already related to the patients disease or genetic mutation (both gene/disease and

Table 4.8: Retrieval Runs with methods focused oncology related trials specifications.

| Run | Study Type | Intervention | Primary Purpose |
|---|---|---|---|
| *Ru_studytype* | x | | |
| *Ru_intervention* | | x | |
| *Ru_primaryporpuse* | | | x |
| *Ru_domain* | x (10) | x (10) | x (10) |
| *Ru_domainBstd* | x (10) | x (20) | x (15) |

demographic searching are always applied on all runs). So these methods are intended to find which trials are focused on cancer treatment and rank them higher.

For this purpose, this section implementation was based on some ideas that were described in the TREC PM 2017 system by [1]. They found that trials were the study type, *ct_studytype*, is *interventional* were more relevant. The rationale behind the dataset is that all patients have malign tumors, so only *intervention* type trials can be useful. Other relevant factor described on [1] is the actual intervention type, *ct_internvention*. Cancer treatment are of three main types, as detailed on chapter 2: *radiotherapy*, *chemotherapy* and *drug related*. The other feature used was the primary purpose, *ct_purpose* that can only be a treatment. Equation 4.5 explains how this fields would be used to query the documents, not using any reference from the patient but only using *ct_fields*: *ct_purpose*, *ct_intervention* and *ct_studytype*.

$$Q_i = (ct\_intervention = true \cup ct\_studytype = true \cup ct\_purpose = true) \tag{4.5}$$

## 4.4 Query Expansion

Query expansion, as the name suggest, consists on adding new synonyms terms to the original query, leading to an increase on possible matching terms to that specific query. For the purpose of this work three types of expansion were used: Pseudo relevance Feedback (PRF), manual expansion and expansion with MeSH terms.

### 4.4.1 Pseudo Relevance Feedback

Pseudo Relevance Feedback (PRF) is a method for automatic local analysis as described on [46] and [67]. It provides possible relevant terms for a new query to the user, without any user intervention. We expanded our queries using the top terms of the "*text*" field, terms with higher TF-IDF scores, from the top retrieved documents on the collection search. Our implementation of PRF query expansions was implemented with Lucene MoreLikeThis (MLT) library. We selected terms from the top-3 retrieved documents and extracting exclusively terms bigger than 4 characters. For example, if our initial query contains the term *"Adenocarcinoma"* (type of lung cancer), we can create a MLT query with terms like, *"lung cancer"*, *"NSCLC"* (Non-Small Cell Lung Cancer). The final query

will contain the initial query, and 3 new expansions, each created from one of the top-3 Documents.

The following are the base parameters that are going to be changed to perform pseudo-relevance feedback:

- Minimum Document Frequency
- Minimum Term Frequency
- Maximum Expansion Length

All runs were tested with BM25, BM25L, TFIDF and LMD. The initial query performed is the run detailed on section 4.3 named $Ru\_domain(BM25L)$ and the final query is named according to the Minimum Document Frequency (MDF) of documents in which the term appears, Minimum Term Frequency (MTF) on the document and Maximum Expansion Length (MEL): $Ru\_qe\_\{MDF\}\_\{MTF\}\_\{MEL\}$. Table 4.9 shows the QE implementations.

Table 4.9: Query Expansion runs parameters.

| Run | MDF | MTF | MEL |
|---|---|---|---|
| *Ru_qe*_10_10_10 | 10 | 10 | 10 |
| *Ru_qe*_10_10_5 | 10 | 10 | 5 |
| *Ru_qe*_25_10_5 | 25 | 10 | 5 |
| *Ru_qe*_25_25_5 | 25 | 25 | 5 |
| *Ru_qe*_50_10_5 | 50 | 10 | 5 |
| *Ru_qe*_50_25_5 | 50 | 25 | 5 |
| *Ru_qe*_50_50_5 | 50 | 50 | 5 |
| *Ru_qe*_50_70_5 | 50 | 70 | 5 |

### 4.4.2 Manual Query Expansion

After the experiments shown on Table 4.8, and analyzing the top-10 retrieved documents for both *Ru_domain* and *Ru_domainB*, two very common definition for "in situ" tumors appear throughout these documents: "Solid tumor" and "Solid neoplasm". So these terms were added to the query and used the same way gene and disease, as Equation 4.6 shows. This idea was already implemented by [33] with proven results on TREC PM 2017. This run was based on *Ru_domain*, and since this was the best run so far after a lot of new retrieval ideas were applied, it would be interesting to see how the search for $tp\_other$ and $tp\_gene$ on *ct_inclusion* affect results. This runs are shown on Table 4.10 as: *Ru_manualexp* using search on *ct_inclusion*, and where domain features consist on using *Ru_domain* search for documents with specific study types, interventions and primary purpose of the trial. *Ru_manualexpNoInc* applies manual expansion and *Ru_domain* leaving out search on

Table 4.10: Retrieval Run with methods focused on manual query expansion.

| Run | Inclusion | Manual Expansion | Domain features |
|---|---|---|---|
| *Ru_domainInc* | x | | x |
| *Ru_manualexpNoInc* | | x | x |
| *Ru_manualexp* | x | x | x |

*ct_inclusion*. Just for control *Ru_domainInc* was created were *Ru_domain* added the search on *ct_inclusion*.

$$Q_i = (tp\_gene \cup tp\_disease \cup "solid\ tumor" \cup "solid\ neoplasm";\ Search\ on : ct\_text) \quad (4.6)$$

### 4.4.3 MeSH Expansion

Most of the disease and genes references present on $tp\_disease$, $tp\_other$ and $tp\_gene$ may be expressed with different terms on relevant documents. Leading to assuming relevant documents as irrelevant because no disease or gene reference were found. To overcome this problem query terms were expanded with synonyms from MeSH, extracting all synonyms with the same level of relevance. For example for the a patient with $tp\_disease$ "lung cancer" a term with the same level of relevance may be "lung neoplasm", on the other hand a less relevant terms is "cancer".

This MeSH expansion was only used on Section 4.5, were ranking lists are re ranked according to the documents *ct_condition*, *ct_meshCondition* and *ct_exclusion*. This technique was also applied on Section 5.2.4 not exactly the same way, but also using MeSH keywords to expand the total number of query terms.

## 4.5 Re-ranking of previously retrieved trials

Not all features can be correctly evaluated using Lucene queries. Thus, filters to work with retrieval system results (ranking lists) were developed, applying functions to Lucene scores for a final re-ranking step. All runs were based on $Ru\_manualexp(BM25L)$. Figure 4.2 shows an example of how both implemented filters work. Green documents have matching condition so condition filter re-ranking will be applied, only increasing the score of matching documents. Red documents have exclusion criteria occurrences, their score will be downgraded using the exclusion filter. Figure 4.3 shows the multiple iterations made from benchmark to benchmark, from the more intuitive methods to more complex and domain specific methods. The diagram doesn't follow a timeline, the flow only show in what order methods were introduced to the system and which implementations make up which benchmarks. The only methods that were not used at retrieval time were: Manual and MeSH expansion that were used before retrieval, and condition re-ranking that was made after retrieval.

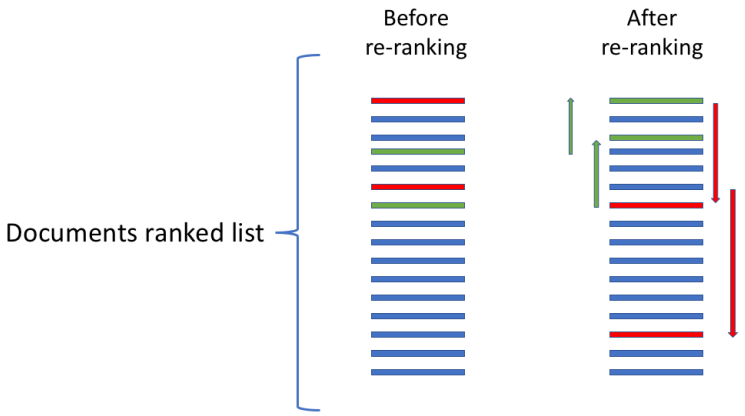


Figure 4.2: Re-ranking methods example. Condition re-ranking examples shown in green and exclusion criteria re-ranking examples in red.

### 4.5.1 Exclusion filter

Two different features were used for this class of methods: the first one utilizes a previously used field, exclusion criteria, but in a completely different fashion. Instead of completely removing documents from the ranking, this method will lower documents scores according to the number of exclusion occurrences.

Filter method start by expanding *tp_other* information for each query and finding synonyms of the disease from MeSH, in order to increase matching probability to the exclusion criteria. These terms are later used for counting matches on *ct_exclusion*. On other hand in order to correctly count the exclusion matches for genes it was also necessary to expand genes with synonyms from MeSH, but also to reduce the specificity of the gene query. For example, gene mutations as "NRAS (Q61K)", extracted from query number 6, gives the system too much accuracy on the gene mutation (amino acid substitution of the position 61 on the NRAS gene [32]). *ct_exclusion* text normally discriminates the gene itself and not the specific amino acid. Thus, using the gene representation present on MeSH definitions instead of "NRAS Q61K" using "NRAS" or "BRAF (V600E)" to simply "BRAF", improved the overall performance of the system.

Besides this simplification, other issue was the towering amount of criteria containing usage references of gene mutations inhibitors. The completeness of the exclusion criteria can lead to misleading matches between patients gene mutations and gene mutations inhibitors as exclusion criteria. After finding matches for a simplified gene mutation in the exclusion criteria, the four following words are checked on *tp_gene* for gene inhibitors usage: Amplifications, translocation, duplication or inhibitor. The ideas implemented on [33], for TREC PM 2017, were a good starting point for using this gene inhibitors. By not counting genes references to inhibitors usage on *ct_exclusion* , one can reduce the number of false negative and false positives. The exception is unless *tp_gene* definition explicitly shows the presence of "Amplifications", "translocation" or "duplication" which

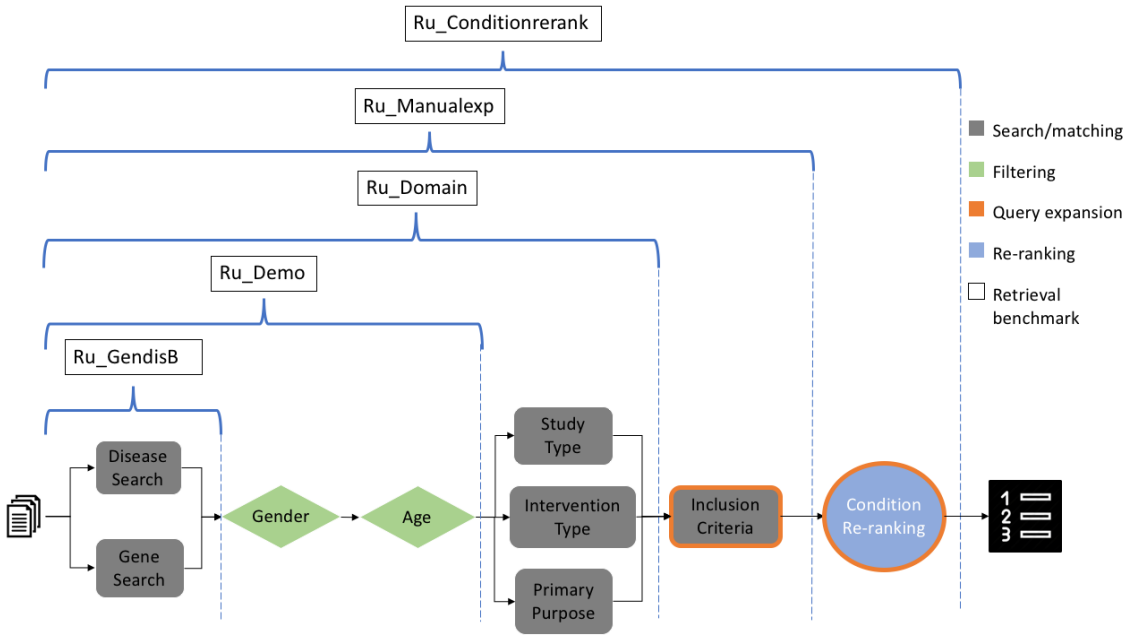


Figure 4.3: Benchmarks for the unsupervised learning system and their respective retrieval methods incremental process.

are associated with inhibitors use.

The re-ranking function is represented in equation 4.7, were $S_x$ is the final score from the retrieval function and $W_x$ is the weight applied manually, and this weight is expected to be the same for an entire ranking list re-ranking.

$$S_x * (1 - W_x * \#occurrences) \tag{4.7}$$

### 4.5.2 Condition filter

The other re-ranking method is based on the official and MeSH conditions present on each clinical trial. After inspecting the top 20 results for each query, it was clear that all relevant documents had the disease or one of its expansions on *ct_condition* or *ct_meshCondition*.

To implement this method, *tp_disease* was first expanded, similar to *tp_gene* and *tp_other* expansions for exclusion criteria, using MeSH terms. This filter will work as a Boolean filter signalling if some of the conditions described on the clinical trial matches patients condition. If the condition present on the query is equal to one of the condition presented on the trial, then this filter will be assigned to "1" and "0" otherwise.

The final score for documents will be given by equation 4.8, were documents have their score increased if their condition feature is "True". Weight $W_x$ is expected to be $0 < W_x \leq 1$, is a constant value for a ranking list re-ranking.

$$S_x * (1 + W_x) \tag{4.8}$$

Table 4.11: Retrieval runs with methods focused on re ranking.

| **Run** | **Inclusion** | **Domain** | **Condition** | **Exclusion re-rank** | **Weight ($W_x$)** |
|---|---|---|---|---|---|
| *Ru_excludererank* | x | x | | x | 0.2/0.1/0.05 |
| *Ru_conditionrerank* | x | x | x | | 0.2/0.1/0.05 |
| *Ru_rerank* | x | x | x | x | 0.2/0.1/0.05 |

Table 4.12: Average term frequency distribution (Relevance judgment documents).

| Terms | Relevance | Text field | Det. Description | Off. Title | Summary |
|---|---|---|---|---|---|
| Genes | Relevant | 4.60 | 1.28 | 0.45 | 0.96 |
| Genes | Not Relevant | 1.22 | 0.36 | 0.08 | 0.20 |
| Disease | Relevant | 8.17 | 2.26 | 1.49 | 2.81 |
| Disease | Not Relevant | 10.16 | 4.19 | 1.30 | 3.07 |

Table 4.13: Average term frequency distribution (top-10 form *Ru_conditionrerank*).

| Terms | Relevance | Text field | Det. Description | Off. Title | Summary |
|---|---|---|---|---|---|
| Genes | Relevant | 8.57 | 1.86 | 0.55 | 1.28 |
| Genes | Not Relevant | 6.34 | 1.88 | 0.40 | 1.30 |
| Disease | Relevant | 7.91 | 2.44 | 1.41 | 2.50 |
| Disease | Not Relevant | 7.40 | 2.65 | 1.10 | 1.95 |

## 4.6 Failure Analysis

In order to methodically develop this thesis and to try to improve existing results, it was indispensable to perform a detailed failure analysis. The goal is to help understand the statistical distribution of relevant terms, and how medical domain references scattered across the clinical trials affect the system performance.

This section is based both on the ranks created by the system developed on this chapter, but also on others implementations for these kind of systems under the same domain. For a more complex analysis, this section is sub divided into two distinct analysis: one for statistical information, and other for domain specific problems faced by the unsupervised learning system.

### 4.6.1 Statistical Information

The statistical analysis is based on the queries terms frequency on the clinical trials, by independently extracting both disease terms frequency and gene terms frequency. What is intended with this analysis is to understand how these features behave according to the clinical trials relevance to the query. To do so, the average frequency of all relevant and non relevant documents for all queries is computed, getting and overall idea of gene and disease terms distribution across the dataset. The following indexed fields where selected: $ct\_text$, $ct\_detailedD$, $ct\_offTitle$ and $ct\_summary$. The average is for all queries as measuring frequency for individual queries may lead to biased results.

Two different group of documents were used to perform this analysis: using all the documents from the relevance judgment file, and using each query top-10 retrieved documents from *Ru_conditionrerank* with BM25L. First, using the relevance judgment an inspection of all the relevant documents was performed. The inspection showed a total of 1132 relevant documents and 11466 non relevant documents. This inspection is not the most specific or fault tolerant, since a lot of non relevant documents are left out. But still, it was considered that most of the non relevant documents that are somehow related to the specific patient are present, so the results presented on Table 4.12 are adequate enough for a global inspection.

The term frequency on the previously detailed case can help understand the dataset discrimination in terms of $tp\_gene$ or $tp\_disease$ distribution throughout relevant and non relevant documents. Table 4.12 shows that even though the disease term is more frequent, non relevant documents appear to have higher count of disease terms. While the gene frequency column shows exactly the opposite: relevant documents have more gene references than non relevant ones. This disease count results clearly corroborate the assumption that most of the non relevant documents for each query on the relevance judgment document (qrels) are somehow related to the query. But, more important to the problem itself, it shows that matching the gene between query and document is significantly more relevant that matching only the disease. This factors also apply when searching for these terms on the full text field, or on the other indexed fields as: *ct_title*, *ct_summary* or the $ct\_detailedD$.

The second statistical group of documents in which the failure analysis was based, was the top-10 documents from *Ru_conditionrerank* with BM25L, the run with one of the best performance on this precision medicine problem. The extraction idea is to understand how the system itself is behaving, and corroborate or not the previous results. Since these top-10 documents were all extracted under the same searching rules, it is natural that the results from relevant and non relevant be more similar than the ones previously extracted from all qrels file. Table 4.13 shows two important results. First, the gene is indeed more relevant since the difference on relevant and non relevant documents in terms of disease terms frequency is very small. In the other hand the gene, clearly provided the system with high levels of precision (accuracy) for the selection of relevant documents. Second, these results show that the system may be giving too much relevance to the disease as a lot of non relevant documents appear to have high counts of disease terms frequency.

The solution to improve the results based on this information can be implemented by increasing the weight of gene searching and decreasing the disease searching relevance. Tuning of the system parameters may increase the performance and subsequently reduce the variance of results, with less false positives on the top ranked documents.

### 4.6.2 Domain failure analysis

For problems as the one faced on precision medicine, it becomes mandatory to use a more detailed and domain specific filtering implementation. Ignoring domain specific techniques means that results will always reach a threshold due to the inability to detect some of the relevant documents that match according to some domain specific criteria. The demographic filter is a good example on how this threshold works: if the exclusion of gender and age was not performed, then a great number of good trials for a given demographic group, will be outperformed by highly relevant trials that don't fit the age and gender of the patient. This way, some relevant documents will always be out of the top results. Ignoring some of the domain specific information is one of the causes of the high level of variance experienced on the results for unsupervised techniques on this chapter. The need for specific filters at searching and re-ranking time is evident after performing two methodical inspections: evaluating the top-10 documents for each query, and understanding the differences between relevant and non relevant documents; evaluating the relevant documents that are out of the relevance zone (out of top-50 documents).

When detailing the top-10 documents for each query, it was evident by the information extracted from the worse performing queries (e.g. 5, 16, 26, 28) that the exclusion criteria is indispensable in order to achieve correct results. This filtering of exclusion criteria is actually more relevant in some queries than others, not that the query itself is harder than the other, but the relevance judgment of the documents leads to a more difficult job in some queries. Nevertheless this filtering should be applied to all queries even to the ones that performed well without it. It was clear after the last implementation using $ct\_exclusion$ for re-ranking, seen on Section 4.5.1, that to correctly understand the exclusion criteria better natural language processing methods should be developed, and can be a good starting point to continue the work developed on this thesis.

For the second type of inspection, the idea was to understand why some documents are ranked so low, given that they're relevant for that given query. Most of the times it was because either $tp\_disease$, $tp\_gene$ or both were scarcely referred throughout the trial. But this is a problem also related to $ct\_exclusion$, this only happens because there are a lot of non relevant documents, that should be excluded, retrieved in front of this relevant documents.

CHAPTER 5

# LEARNING TO RANK WITH TREES

Information retrieval techniques as seen on chapter 4 can deal with categorical or numeric field filtering, as age or gender exclusion. But in cases where a more detailed and precise filtering of clinical trials is needed, these IR techniques may not have the flexibility to solve this problem, with the required precision. As shown on section 6.4, some of the queries (e.g. 15, 16, 26 and 28) are poorly answered. The next step in order to overcome some problems of the IR methods, is to use supervised learning techniques. More specifically, solving learning to rank problems, which predict the ranking of documents for a given query. These learning to rank systems are made up by two different sub systems: learning system and ranking system [40]. Table 5.1 explains all used notations throughout the chapter.

Table 5.1: Notation used in this thesis.

| Notation | Definition |
|---|---|
| $ct$ | Clinical Trial |
| $Rs$ | A set of methods used to create a rank list with supervised learning |
| $Rs\_algorithm$ | Retrieval run. Algorithm detail the algorithm used on the run. |
| $ct\_field$ | *Field* content given Clinical Trial $ct$. |
| $tp\_field$ | Topic (Query) *field* content. |

## 5.1 Decision trees on Clinical Trials

When using medical features with unsupervised techniques, one of the most difficult jobs is to correctly weight the relevance of features for the query search, leading to the

question: which one of the medical features should be more relevant for the query? And, is it necessary that all match the expected results? The three main problems of using unsupervised learning for this types of problems are that: Medical features extracted can be irrelevant, each feature have different importance and Individually weighing these features can be a very hard and time consuming job. Supervised learning models can overcome this problem because it learns the training data information, so the assumptions made about features are entirely based on correct information.

The types of limitations described previously for unsupervised learning can be overcome with supervised learning, because the learning module of the system will understand how the feature vector of a relevant document can be composed, and weight all features according to their correlation to relevant clinical trials. Which are not necessarily the assumptions made by the developer, when creating the unsupervised learning system, so even if irrelevant features are extracted the system has the ability to understand it and reduce their importance on the learned model.

Even doe this problem could be solved with many different machine learning structures, for this thesis the only structure used for supervised learning was decision trees. But why using decision trees and not other structures? The two main reasons are: First they are non-linear decision algorithms. As the name suggest, being a non-linear algorithm means that data does not follow a linear equation. This is easy to understand if we look at the Figure 5.1 left graph showing a linear regression. All prediction made assume that, as a certain feature increases or decreases the other features follow as expected according to a linear equation ($F(x) = mx + b$), so features are dependent of each other. On non-linear algorithms features are independent of each other so their equation is non-linear, which is the case with the features from the problem in hand and exactly as shown on the right graph of Figure 5.1, were a decision tree regression is made. The second reason is the induction task which is used to create these tree like structures, as detailed on [55]. Such model simplifies decision interpretability, giving a more human-like structure to algorithms users, were each node represents a certain features and one of many possible thresholds for that feature, as Figure 5.2 shows and as explained on [12].

## 5.2 Clinical trials features

This chapter will be dedicated to supervised learning techniques for solving ranking problems. It details how one can make better use of these supervised learning assumptions and how can we make them specific for the clinical trials data used on this work. To do so, the chosen algorithms are based on decision trees models and multiple applications of ensemble methods. The task performed by these trees, induction on [55], is ideal for the types of queries we are facing in clinical decisions. The trees create boundaries and rules within their nodes, as explained on [12], to differentiate the documents either by performing regression, classification or ordinal regression.

While parsing clinical trials, the extraction of annotation fields (i.e. fields containing

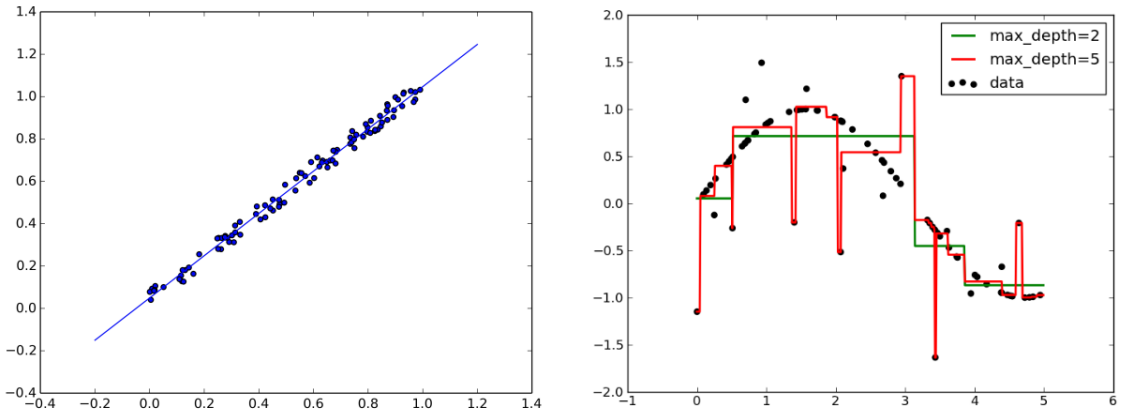


Figure 5.1: Linear vs non-linear algorithms. Left graph shows a linear regression extracted from [61] and the right graph shows a decision tree regression extracted from [54].

simple annotations instead of descriptive text) is very important, since, if the correct information is extracted, these systems can create boundaries for filtering documents. These can be used to filter documents according to demographic information, to oncology related information and exclusion and inclusion criteria. Figure 5.2 shows a good example on how can these boundaries help the system differentiate each new instance. Each one of the documents goes through this structure and is given a final value accordingly to the documents relevance to the query, so that documents can be ordered by their relevance. Final scores and BM25 score threshold are just examples of values to better represent how the structure works.

Performing this type of earning with decision trees, for cases as a clinical trial information extraction is becoming a commonly applied practice. Chapter 3 shows some examples of its usage, such as: [40, 42, 43], all three containing multiple implementations of learning to rank algorithms with decision trees, showing features used, how they were used and results of all used algorithms.

### 5.2.1 Features from the Clinical trials

Before performing any Learning to Rank, its mandatory to have a good set of feature vectors that describe documents as thorough as possible. These features must be a mixture of medical related features, described on 5.2.2, and statistical features extracted by applying multiple information retrieval algorithms (e.g. BM25 (L,+), LMD, LMJM).

From the clinical trials dataset 58 features were created, as shown in Table 5.2. In order to achieve even better results, continuous valued features were individually tested by ranking each individual feature to later calculate their individual precision at five, ten, twenty and thirty, Table 5.2 show these results for P@10. For reading full results of this feature selection go to Table A.5, which shows all values of precision for each feature, ordered by their precision at five (the primary metric in which the selection of feature was based), showing which are the best features and worst feature to be used on the supervised

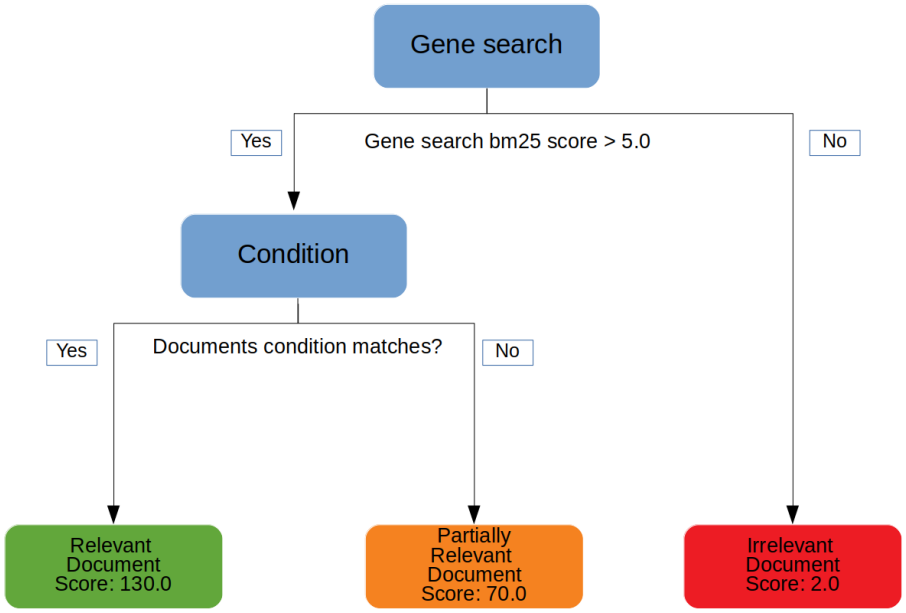


Figure 5.2: Example of a decision trees structure used for ordinal regression on the oncology domain.

learning system.

Table 5.2: All features used for learning to rank with TREC PM 2017 (28 queries).

| ID | Feature Description | Precision@10 |
|---|---|---|
| 1 | Primary Purpose of document (Boolean) | – |
| 2 | Intervention type (Boolean) | – |
| 3 | Study type (Boolean) | – |
| 4 | Trial Condition check (Boolean) | – |
| 5 | Inclusion criteria references count | – |
| 6 | Exclusion criteria references count | – |
| 7 | Gene Keywords count on full text | 0.23928571 |
| 8 | Gene Keywords count on detailed description | 0.225 |
| 9 | Gene Keywords count on official title | 0.20714286 |
| 10 | Gene Keywords count on brief summary | 0.17142858 |
| 11 | Gene Keyword percentage on full text | 0.24285714 |
| 12 | Gene Keyword percentage on detailed description | 0.225 |
| 13 | Gene Keyword percentage on official title | 0.071428575 |
| 14 | Gene Keyword percentage on brief summary | 0.060714286 |
| 15 | Disease Keywords count on full text | 0.075 |
| 16 | Disease Keywords count on detailed description | 0.08214286 |
| 17 | Disease Keywords count on official title | 0.11071429 |
| 18 | Disease Keywords count on brief summary | 0.075 |

Table 5.2 – continued from previous page

| ID | Feature Description | Precision@10 |
|---|---|---|
| 19 | Disease Keyword percentage on full text | 0.08928572 |
| 20 | Disease Keyword percentage on detailed description | 0.09642857 |
| 21 | Disease Keyword percentage on official title | 0.08214286 |
| 22 | Disease Keyword percentage on brief summary | 0.071428575 |
| 23 | BM25L score for disease and gene search on full text | 0.03214286 |
| 24 | BM25L score for disease and gene search on detailed description | 0.075 |
| 25 | BM25L score for disease and gene search on official title | 0.08214286 |
| 26 | BM25L score for disease and gene search on brief summary | 0.071428575 |
| 27 | BM25 score for disease and gene search on full text | 0.03214286 |
| 28 | BM25 score for disease and gene search on detailed description | 0.075 |
| 29 | BM25 score for disease and gene search on official title | 0.08214286 |
| 30 | BM25 score for disease and gene search on brief summary | 0.071428575 |
| 31 | BM25+ score for disease and gene search on full text | 0.03214286 |
| 32 | BM25+ score for disease and gene search on detailed description | 0.075 |
| 33 | BM25+ score for disease and gene search on official title | 0.28214285 |
| 34 | BM25+ score for disease and gene search on brief summary | 0.175 |
| 35 | LMD score for disease and gene search on full text | 0.2 |
| 36 | LMD score for disease and gene search on detailed description | 0.19642857 |
| 37 | LMD score for disease and gene search on official title | 0.042857144 |
| 38 | LMD score for disease and gene search on brief summary | 0.07857143 |
| 39 | LMJM score for disease and gene search on full text | 0.05 |
| 40 | LMJM score for disease and gene search on detailed description | 0.05357143 |
| 41 | LMJM score for disease and gene search on official title | 0.19285715 |
| 42 | LMJM score for disease and gene search on brief summary | 0.14642857 |
| 43 | TFIDF score for disease and gene search on full text | 0.29642856 |
| 44 | TFIDF score for disease and gene search on detailed description | 0.22857143 |
| 45 | TFIDF score for disease and gene search on official title | 0.15 |
| 46 | TFIDF score for disease and gene search on brief summary | 0.10357143 |
| 47 | TF score for disease and gene search on full text | 0.225 |
| 48 | TF score for disease and gene search on detailed description | 0.13571429 |
| 49 | TF score for disease and gene search on official title | 0.13214286 |
| 50 | TF score for disease and gene search on brief summary | 0.017857144 |
| 51 | full text length | 0.17857143 |
| 52 | detailed description length | 0.08928572 |
| 53 | official title length | 0.08928572 |
| 54 | brief summary length | 0.1 |
| 55 | IDF score for disease and gene search on full text | 0.092857145 |
| 56 | IDF score for disease and gene search on detailed description | 0.08928572 |

Table 5.2 – continued from previous page

| ID | Feature Description | Precision@10 |
|---|---|---|
| 57 | IDF score for disease and gene search on official title | 0.11071429 |
| 58 | IDF score for disease and gene search on brief summary | 0.18928571 |

### 5.2.2 Medical Related Features

As shown on Chapter 4, medical related information can be very helpful for boosting the performance of IR systems when ranking documents. Though most of these features are scattered across unstructured clinical trials texts (e.g brief summary, detailed description), and may produce incorrect assumptions when searching for medical terms (e.g. radiology, chemotherapy) on text fields. These features are based on document fields that serve as guidelines (annotations) for the intervention detailed on the clinical trial: Primary purpose, Intervention and Study type. These three features will be used as binary features, depending on whether the fields correspond to cancer related assumptions or not. They help the system perform a more human-like dissection of the information within the document, understanding with more precision the documents relevance for any kind of cancer related condition, and not the patient specific condition.

One last binary feature was created, this time representing the specific condition the clinical trial is focused on, and whether it matches the patient condition. Using exactly the same implementation used for unsupervised learning on the condition filter for re ranking detailed on section 4.5.2. These features are represented on Table 5.2 from feature 1 to 4.

### 5.2.3 Extracting Exclusion criteria from CT Narrative

After inspecting a couple of relevant documents and the top 10 retrieved documents for the queries with the worst individual precision (e.g. 5, 26, 27), it becomes clear that one of the obvious reason for this bad performance is the incorrect use of *ct_exclusion*. The high variance in retrieval performance found among the answered queries may be correlated to the importance of *ct_exclusion* field on a given document. Queries who have more documents were *ct_exclusion* is indispensable for the document relevance assertion, tend to be worse answered than queries with less important information on *ct_exclusion* .

It was clear that developing a system capable of understanding the medical literature in a more human fashion (i.e. understanding what term was actually intended to be excluded on a sentence or if the sentence is on positive or negative form (e.g. Gene BRAF is not to be excluded)), was a job out of range for this work. So in order to understand if this was a problem where decision trees could lead to new ways of understanding the data and more accurately, infer which documents are to be excluded or not. We used the methods described in the unsupervised learning re-ranking, section 4.5.1, as features. Feature development was not only focused on exclusion but in a symmetric way applied to

inclusion, counting both occurrences of $tp\_other$, $tp\_gene$ and their respective expansions on $in\_exclusion$ and $in\_exclusion$. These features are represented on Table 5.2 by both features 5 and 6.

### 5.2.4 Automatic keywords matching

Clinical trials contain fields with keywords extracted from MeSH database, that serve as a quick and easy summary of the intervention and acceptable patients characteristics. Unfortunately, not all documents contain them or existing keywords may be incomplete. In addition to these provided terms, the system developed in this thesis is capable of automatically extract MeSH keywords from text, using keywords lists from three different categories within MeSH database: "Diseases" (C), "Chemicals , Drugs" (D) and "Analytic, Diagnostic and Therapeutic Techniques and Equipment" (E). This implementation was based on Novasearch runs for TREC CDS 2015 [50]. This enables extracting disease information, names of drugs and other chemicals used for treatment, and the interventions actually performed (detailed) withing the clinical trial text.

Extracting keywords from documents smaller text fields (e.g. $ct\_offtitle$) may lack precision, because descriptions not always detail the specific disease or gene for the trial. On the other hand longer text fields (e.g. $ct\_detailedD$) can create unrelated keywords, because the clinical trials text may refer other diseases and genes mutations as example or for comparisons. For this two motives, the keywords are extracted from the query and then matched to all the text fields on the indexed documents, and not the other way around. For example: on the gene "IDH1 (R132H)", (R132H) details the gene mutation too strictly for what the relevant documents actually describe about the gene, and so the keyword "IDH1" is added.

The implementation extracts MeSH keywords, creating two keyword lists for each query: one for genes and the other for the diseases. The terms on the collections are added as keywords when

- The keyword is equal to the query term or is contained on it;
- The keyword contains the first two query terms together (also adding both terms individually if terms are more than three characters long);
- The keyword contains the first query term individually, only the first query is added.

After some preliminary experiments, we observed that no irrelevant (low accuracy) list of keywords was created. For example on query "Pancreatic adenocarcinoma", a irrelevant list of keywords would be the three following lists: ["pancreatic"], ["adenocarcinoma"] or ["pancreatic","adenocarcinoma"]. Where no composed keyword is extracted for a multi-term query, which can lead to high scores for non relevant documents.

Matching these keywords and documents is not trivial. For example, automatically creating the keywords for $tp\_disease$ "Lung Cancer", creates three keywords: "lung", "cancer" and "Lung cancer". The frequency of keywords on the documents was used to get

keywords and documents co-relation in a quantitative manner. It was also used the percentage of keywords from the list that were found on the document, to get their co-relation in a qualitative way. Since keyword lists with composed keywords also contains the individual term, for example if a list contains "lung cancer" it also contains "lung" and "Cancer". This technique ensures that if a text doesn't contain a high % of the keyword terms, the keyword that is missing is always a relevant keyword. Following from the previous example of a keywords list, if not 100% of the keywords are found then at least one of the missing keywords is the composed one, "lung cancer". These features are shown on Table 5.2 from feature 7 to 22.

## 5.3 Learning-to-Rank algorithms

As described on chapter 3, all algorithms used on this thesis are based on a specific machine learning model, decision trees. The goal is to ensure more methodical and controlled experiments, helping on the comparison of multiple machine learning approaches to learning-to-rank (e.g. pairwise, listwise), and the effect of ensemble methods on ranking problems even when, as the TREC PM 2017, we have a small amount of queries available to train and test the system. The following algorithms explanations are all detailed according to the papers in which they were presented to the scientific community, or papers exclusively focused on them, so some of the ideas presented on this section are adapted from the following papers: [11, 14, 66, 68].

### 5.3.1 LambdaMART

LambdaMART is a combination of two machine learning algorithms, lambdaRank and MART as detailed on [14]. LambdaRank is a neural network ranking model, where the idea is to use a cost function that minimizes the number of incorrect order pairs in the results. But instead of applying this to a simple cost function that utilizes a single point (as RankNet), lambdaRank uses the gradient of the cost function of a pair of points as the cost function, as detailed on [66]. This way, when optimizing the cost function, two points will be used at once, increasing and decreasing their ranks while leaving the other points untouched. MART (Multiple Additive Regression Trees) is a gradient boosting decision trees model, that combines multiple "weak" learners outputs to create a final strong learner. It is based on the adaptative boosting ensemble method, explained on section 3.3.4. Given the need for optimization of a cost function, the combination of these learners is going to build up iteration after iteration to create a strong learner. Algorithm 1 shows a top level idea on how lambdaMart works.

### 5.3.2 AdaRank

AdaRank is the only listwise algorithm used on the implementation of this work, and it is based upon the AdaBoost (adaptive boosting) algorithm. As follow from [68], paper

```
Data: Samples (each sample is a ranked pair of documents for a given query)
Result: Trained model ready for test
Parameter: nTrees (number of trees parameter), nTreeLeaves (number of leaves
           parameter), learningRate
for i = 0; i < nTrees; i = i + 1 do
    Learning Phase
    Compute Lambda (for each training sample lambda serve as a training label);
    rt = RegressionTree(nTreeLeaves, samples)
    TreeModel = rt.fit;
    leaves = TreeModel.leaves;
    for k = 0; k < leaves.length; k = k + 1 do
        leavesSamples = leaves[k].samples;
        for l = 0; l < leavesSamples.length; l = l + 1 do
            score[leavesSamples[l]] =
              score[leavesSamples[l]] + learningRate * leaves[k].output;
        end
    end
    Validation phase
    for k = 0; k < modelScoresOnValidation.length; k = k + 1 do
        for j = 0; j < modelScoresOnValidation[k].length; j = j + 1 do
            modelScoresOnValidation[k][j] = modelScoresOnValidation[k][j] +
              learningRate * rt.featureEval(validationSamples[k][j]);
        end
    end
end
```

**Algorithm 1:** LambdaMART algorithm detail. Extracted from [25] implementation of LambdaMART algorithm.

detailing the algorithm, it utilizes a specific type of boosting: instead of optimizing the cost function, this algorithm change the weight of misclassified points in a adaptive manner. This makes the classification of points with higher weights, a priority for the learner. The difference is that AdaRank, being a listwise approach, weights the entire list (ranking list for a query) instead of increasing the weighting of individual points. Each "weak" learner represents a given ranking list for a specific query. At each iteration, the algorithm changes the weights of the queries lists with the purpose of optimizing a set IR metrics (e.g. NDCG, Precision).

This algorithm takes instances as ranking lists, avoiding known problems for pairwise and pointwise approaches such as document pairs for the same query that are not independently distributed. Training on the ranking list top results is unfeasible with both pointwise and pairwise, since the document pairs will be from any part of the list. This problem does not occur in listwise approaches since the ranking list will be affected on places were relevant swaps on the ranking will affect the IR measure, normally is the top of the ranking list. Listwise algorithms are also unaffected by the number of document pairs vary from query to query, which may lead to biased results towards queries with more documents on point or pairwise algorithms. Algorithm 2 shows how the Adarank

```
Data: Samples (each sample is a RankList of documents)
Result: Trained model ready for test
Parameter: nIterations (number of iterations), max (maximum number of round
           without improvement)
bestModelScore = 0
for i = 0; i < nIterations; i = i + 1 do
    Learning Phase
    WeakRanker = WeakRankerLearner
    for RankList Sample: Samples do
        tmp = Score(Rank(Sample))
        trainedScore = trainedScore + tmp
    end
    if trainedScore! = lastTrained then
        UsedFeatures = null
    else
        if lastFeature == WR.BestFeature then
            consecutive = consecutive + 1
            if consecutive == max then
                UsedFeatures = null
            end
        else
            UsedFeatures = null
            consecutive = 0
        end
    end
    Validation Phase
    if ValidationSamples! = null then
        score = Score(validationSamples)
        if score > bestModelScore then
            bestModelScore = score
        end
    end
end
```

**Algorithm 2:** AdaRank algorithm detail. Extracted from [25] implementation of the AdaRank algorithm.

algorithm behaves and how it created the model to be used for ranking creation.

### 5.3.3 Random Forests

This algorithm was developed by [11] and is based on ensemble ideas to overcome limitations of decision trees models, such as the tendency to overfitting the data at learning time. More specifically, this algorithm is very similar to the ensemble procedure called bootstrap aggregation or bagging. Bootstrapping samples are random generated samples, created by sampling points uniformly and with replacements (i.e. duplicate values will appear on the samples) from an initial training set D, creating multiple new training sets based on D. These sets are later aggregated by selecting the mode when dealing with classification problems, and mean average for regression problems.

**Data:** Samples (each sample is a ranked pair of documents for a given query)
**Result:** Trained model ready for test
**Parameter:** nBags (number of bags parameter), nTrees (number of trees parameter), nTreeLeaves (number of leaves parameter), learningRate
Learning phase
**for** $i = 0;\ i < nBags;\ i = i + 1$ **do**
  $bag = samples\ randomly\ sampled\ from\ the\ training\ set$
  $Model = new\ LambdaMartModel(nBags, nTrees, nTreeLeaves, learningRate)$
  $Learn(Model)$
**end**
Validation phase
**if** $ValidationSamples! = null$ **then**
  $scoreOnTraining = Score(rank(samples))$
  $score = Score(validationSamples)$
**else**
  $scoreOnTraining = Score(rank(samples))$
**end**

**Algorithm 3:** Random Forest algorithm detail. Extracted from [25] implementation of the Random Forest algorithm.

On [11] we have an example of an random generated vector of features in Random forests called random subspace method or feature bagging. For each new training sample, it creates a random subset of the initial set of features. This process increases the correlation between the features that are better predictors, creating the model with a better estimation of which features are better for the classification/regression task. This random use of the dataset is expected to improve the results of variance on the training set, subsequently reducing one of the biggest problems with decision trees, over-fitting. The implementation of the algorithm used was based on lambdaMART algorithm, so it is expected that the results of random forests are better than lambdaMART original results. This ensemble methods can be a good solution to overcome our limitation in terms of number of queries. Algorithm 3 shows how the ensemble based algorithm random forests workS.

## 5.4 Baselines for supervised Learning

To benchmark supervised learning implementation, we tested LambdaMART, AdaRank and Random Forests from the RankLib Library, a library specialized on out of the box learning to rank algorithms. RankLib gives users access to a lot of features and parameters to dynamically correlate the algorithm to anykind of dataset. This library utilizes IR measures as a way to compare and optimize the models created.

Contrariwise to what would be a natural choice, precision was not used as training metric. Precision can only tell us the percentage of relevant documents withing a certain ranking (e.g. 5, 10, 20, 30), and doesn't provide any insight in terms of what documents come first on the ranking. Lets assume we trained a model with precision at 10. Such model could reliably retrieve a considerable number of relevant documents, but always

Table 5.3: RankLib Baselines

| Baseline | Algorithm | IR metric |
|---|---|---|
| *Rs_lambdaM* | LambdaMART | NDCG@10 |
| *Rs_randomF* | Random Forests | NDCG@10 |
| *Rs_adarank* | AdaRank | NDCG@10 |

rank them out of the top-5, while the top-5 is full of non relevant documents.

On the other hand, NDCG, which is explained on Chapter 6, takes into account the documents actual ranking, so the best model would be the one that continuously retrieves more relevant documents with a higher rank. The baselines shown on Table 5.3 were all implemented after performing cross validation for parameter tuning as explained on Section 6.3.2 on the following chapter.

# CHAPTER 6

# EVALUATION

This chapter results are based on implementations for the TREC PM 2017, as [1], and multiple systems based on other retrieval tasks as [50] and [49], NovaSearch team systems for TREC CDS 2015 and 2014 respectively. Using both implementations and results of each system to compare this thesis implementations results and ideas. Throughout this chapter all evaluation metrics, experiments and results will be summarized, analyzed and explained in order to perceive the quality of the developed systems. Showing the actual benefits or not of each individual retrieval method.

## 6.1 Methodology

### 6.1.1 Dataset

TREC PM 2017 clinical trials dataset was extracted from clinicaltrials.gov, and provided in a semi-structured way subdivided in fields. The fields extracted and used on this work are the ones detailed on table 4.3. The following listing shows an example of a clinical trial from the dataset used:

Listing 6.1: Example of a clinical trial from the TREC PM 2017 dataset

```
1 <brief_title>Additional Benefit of Cilostazol to Dual Antiplatelet Therapy
     After Biolimus-eluting Stent Implantation</brief_title>
2   <acronym>ABCD</acronym>
3   <official_title>A Trial of Evaluating Additional Benefit of Cilostazol to
       Dual Antiplatelet Therapy in Patients With Long or Multi-vessel Coronary
       Artery Disease Underwent Biolimus-Eluting Stent Implantation</
       official_title>
4   <sponsors>
5     <lead_sponsor>
6       <agency>Yonsei University</agency>
```

The topics (clinical reports) used to query the system were all from oncology patients. They are divided into four fields: disease, gene, demographic and other. The disease is the actual disease the patient suffer from. The gene mutation the patient suffers from is under the field "gene". Demographic represents the age and gender of the patient with no personal information provided. "Other" field is all the other conditions the patient suffers, that may be relevant for physicians. All the queries used for this work are detailed on table A.4.

### 6.1.2 Metrics

For this work the metrics used for evaluation were Precision at (5, 10, 15, 20), Recall and Normalized Discounted Cumulative Gain, all extracted from [46]. As the names suggests, TREC Precision Medicine (PM) 2017 is a system in were the principal metric utilized for evaluation was precision. Precision is represented by the following formula:

$$Precision = \frac{RelevantDocs \cap RetrievedDocs}{RetrievedDocs} \tag{6.1}$$

It is the relationship between the number of relevant documents within the number of documents retrieved. So for example, P@5 represents the ratio of relevant documents on the top 5 retrieved documents.

Recall measures the number of relevant documents retrieved from the total of relevant documents:

$$Recall = \frac{RelevantDocs \cap RetrievedDocs}{RelevantDocs} \tag{6.2}$$

Recall can also be measured at specific number of documents retrieved (e.g. 5, 10, 15, 20). We used the total number of relevant existing on the dataset and total of relevant retrieved.

Discounted Cumulative Gain is a commonly used measure to evaluate ranking quality. It measures the ranking effectiveness of relevant documents, relative to each other. This metric goes by the following equation 6.3:

$$DCG_p = \sum_{i=1}^{p} \frac{2^{rel_i} - 1}{\log_2(i+1)}, rel \in 0, 1 \tag{6.3}$$

$p$ is the total number of retrieved documents, $rel$ is a binary variable with value 0 when the document $i$ is not relevant and 1 otherwise. Ideal Discounted Cumulative Gain is the particular case where the $n$ relevant documents retrieved are on the first $n$ positions of the ranking. So IDCG is the maximum possible score with $n$ relevant retrieved documents.

$$IDCG = \sum_{i=1}^{|rel|} \frac{2^{rel_i} - 1}{\log_2(i+1)}, rel \in 0, 1 \tag{6.4}$$

The normalization 6.5 or NDCG is the relation between Discounted Cumulative Gain and the Ideal Discounted Cumulative Gain.

$$nDCG = \frac{DCG}{IDCG} \tag{6.5}$$

## 6.2 Experiments and Results: unsupervised learning

The results were divided by the different techniques utilized, including results for multiple information extraction techniques, the usage of QE and for the re-ranking experimnets. All the results are fully detailed by tables on Section A.

### 6.2.1 Unsupervised learning Protocol

The first step is the index creation, as explained on section 4.2. Where all the fields detailed on Table 4.3 are extracted from the semi-structured documents, and processed to make up the index. To retrieve documents from the dataset, the query also needs to be processed, using the same analyzer as the index. After query terms assessment is made, multiple techniques were applied for searching on each of the indexed text fields, filtering with other fields (e.g. age, gender), or applying query expansion and rank fusion algorithms.

To test the system relevance judgment files and a system called trec_eval was used. The relevance judgment files serve as "ground truth" providing some of the non relevant and all the relevant documents for each of the topics represented on Table A.4. The relevance goes on a scale from 0 to 2, where 0 is non-relevant, 1 is partially relevant and 2 definitely relevant for the query. The trec_eval evaluation system, is a system capable of inferring multiple types of metrics(e.g. the ones referred on 6.1.2) for a ranking list, with options for average results over multiple queries or individual queries results.

### 6.2.2 Retrieval Function Comparison

The first results on table 6.1, show a clear distinction between TFIDF and the rest of the models. TFIDF had much worse results when compared to other retrieval functions. The LMD model did not performed as well as BM25 or BM25L with the demographic filter but produced very good results when using the oncology specific filter(see table A.8). BM25 and BM25L are good solutions for ranking functions for this problem, but given the towering amount of long documents found on the corpus, BM25L achieved even better results, and was seen as the best retrieval function for this problem.

### 6.2.3 Demographic and Domain Specific Filters

From the results on Table 6.1, one can see that both demographic filters increase the precision (P@5, P@10 and P@15), NDCG and recall metrics. The IR system that utilizes both filters together (age, gender filters) gets the best results, meaning that demographic filtering helps to return better documents. In terms of retrieval functions, BM25L achieved the best overall scores in all baselines, since the documents we are dealing are considered very long. It is closely followed by BM25 and LMD while TFIDF was the worse performing technique.

Table 6.2 shows the results for inclusion and exclusion criteria, where both "other" and "gene" were used to create queries with inclusion and exclusion criteria. The introduction

Table 6.1: Demographic filter results.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| *Ru_gendisB* (BM25L) | 0.4500 | 0.3714 | 0.3405 | 0.5375 | 0.7563 |
| *Ru_demo* (BM25L) | **0.4714** | 0.4000 | **0.3667** | **0.5455** | **0.7588** |
| *Ru_gendisB* (BM25) | 0.4286 | 0.3679 | 0.3405 | 0.5297 | 0.7496 |
| *Ru_demo* (BM25) | 0.4500 | 0.3964 | 0.3643 | 0.5403 | 0.7528 |
| *Ru_gendisB* (LMD) | 0.4143 | 0.3857 | 0.3214 | 0.5067 | 0.7470 |
| *Ru_demo* (LMD) | 0.4286 | **0.4036** | 0.3333 | 0.5154 | 0.7464 |
| *Ru_gendisB* (TFIDF) | 0.3643 | 0.3357 | 0.3000 | 0.4886 | 0.7083 |
| *Ru_demo* (TFIDF) | 0.3714 | 0.3321 | 0.2929 | 0.4615 | 0.6937 |

Table 6.2: Exclusion criteria results.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| *Ru_demo* (BM25L) | **0.4714** | 0.4000 | 0.3667 | 0.5455 | 0.7588 |
| *Ru_exclusion* (BM25L) | 0.4343 | 0.3964 | 0.3643 | 0.5165 | 0.7047 |
| *Ru_inclusion* (BM25L) | 0.4657 | **0.4107** | **0.3714** | **0.5500** | **0.8003** |
| *Ru_criteria* (BM25L) | 0.4143 | 0.3964 | 0.3452 | 0.4721 | 0.6668 |
| *Ru_demo* (LMD) | 0.4286 | 0.4036 | 0.3333 | 0.5154 | 0.7464 |
| *Ru_exclusion* (LMD) | 0.4357 | 0.3893 | 0.3524 | 0.5544 | 0.7908 |
| *Ru_inclusion* (LMD) | 0.4286 | 0.3857 | 0.3333 | 0.4893 | 0.6879 |
| *Ru_criteria* (LMD) | 0.4071 | 0.3571 | 0.3238 | 0.4582 | 0.6541 |

of exclusion criteria decreased retrieval performance for both BM25 and BM25L on all metrics, probably caused by an increase on the number of relevant documents excluded by a misleading match between additional diseases and exclusion criteria. LMD was the only similarity used that takes into account the inter relationship of the terms on the documents, and was the only which increased the performance of the system using the exclusion criteria. So term inter relationship can be very helpful for future work on the exclusion criteria. On the other hand inclusion decreased on precision at 5 (P@5) but all other metrics increased when compared to *Ru_demo*, since the optimal ranking list should have the best score on P@5 this method was discarded for now.

Using the fields Study Type, Primary Purpose and Intervention greatly increased the performance of the system, This process increases on the count of cancer related documents returned. The results can be seen on Table 6.3 for BM25L retreival function. Using these fields helps increasing the score of cancer related documents, as explained on 3.4.1. The last and best experiment within this domain related features was manual query expansion by adding "solid tumor" and "solid neoplasm" to the query on all text information within the trial. Since this was the best run so far before performing any type of re-ranking, the previously used method for search on the inclusion criteria was re-used and leveraged the results in all metrics, this showing how important it is to create runs in

Table 6.3: Oncology filter results.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| *Ru_demo* (BM25L) | 0.4714 | 0.4000 | 0.3667 | **0.5455** | **0.7588** |
| *Ru_domain* (BM25L) | 0.5214 | 0.4250 | 0.3762 | 0.5108 | 0.6864 |
| *Ru_domainBoosted* (BM25L) | 0.5214 | 0.4179 | 0.3714 | 0.4938 | 0.6443 |
| *Ru_domainInc* (BM25L) | 0.5000 | 0.4393 | 0.3762 | 0.5000 | 0.7550 |
| *Ru_manualexpNoInc* (BM25L) | 0.5286 | 0.4321 | 0.3548 | 0.5187 | 0.6722 |
| *Ru_manualexp* (BM25L) | **0.5500** | **0.4500** | **0.3976** | 0.5243 | 0.7175 |

Table 6.4: Results for the re ranking implementation.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| $Ru\_excluderer ank(BM25L)$ | 0.5571 | 0.4357 | 0.3905 | 0.5220 | 0.7175 |
| $Ru\_conditionrerank(BM25L)$ | **0.5643** | **0.4607** | **0.4119** | **0.5331** | **0.7175** |
| $Ru\_rerank(BM25L)$ | 0.5571 | 0.4500 | **0.4119** | **0.5301** | **0.7175** |

a methodically and iterative manner.

This experiment achieved very good results and matches the best team on TREC PM 2017 P@5 results. For more results go to Table A.6, Table A.8 and Table A.7 on appendix. These tables show all implementations results for all retrieval functions.

### 6.2.4 Re-ranking experiments

Re ranking implementations were all based on *Ru_manualexp*, the best retrieval baseline so far. Results are really positive, specially when applying the condition filter, as table 6.4 shows. The other two baselines also increased *Ru_manualexp* results, shown on Table 6.3, which shows that re ranking is a very good practice after performing the initial retrieval process. *Ru_excluderer ank* best results used a weight(W(x)) of 0.05, meanwhile *Ru_conditionrerank* best run used a W(x) of 0.1.

### 6.2.5 Pseudo Relevance Feedback Experiments

As shown on table 6.5, with the best PRF implementations, query expansion had a small or no increase on the performance of the system. The expansion of new words from the actual dataset led to a small increase on P@5 results with some baselines but decrease all other precision metrics in all other runs for QE. Applying query expansion to $Ru\_domain(BM25L)$, only had a little increase on P@15 results in some QE baselines. Check table A.9 to see the full set of detailed results.

Since results on query expansion were not as expected, two options could be taken:

- expand the query with specific words from external sources (i.e. queries are based on very precise genes and disease so expansion terms may be less specific for the topic than the initial terms if extracted from the dataset);

Table 6.5: Results for the query expansion technique.

| **Baseline** | **P@5** | **P@10** | **P@15** | **NDCG** | **Recall** |
|---|---|---|---|---|---|
| *Ru_domain* (BM25L) | **0.5214** | **0.4250** | 0.3762 | **0.5108** | **0.6864** |
| *Ru_qe*_10_10_10 (BM25L) | **0.5214** | 0.3750 | 0.3119 | 0.4780 | 0.6337 |
| *Ru_qe*_10_10_5 (BM25L) | 0.5143 | 0.3857 | 0.3405 | 0.4881 | 0.6410 |
| *Ru_qe*_25_10_5 (BM25L) | 0.5143 | 0.3929 | 0.3405 | 0.4864 | 0.6410 |
| *Ru_qe*_25_25_5 (BM25L) | 0.4714 | 0.4036 | 0.3595 | 0.4904 | 0.6461 |
| *Ru_qe*_50_10_5 (BM25L) | 0.5143 | 0.3929 | 0.3405 | 0.4803 | 0.6401 |
| *Ru_qe*_50_25_5 (BM25L) | 0.4714 | 0.4071 | 0.3595 | 0.4898 | 0.6454 |
| *Ru_qe*_50_50_5 (BM25L) | **0.5214** | 0.4179 | 0.3738 | 0.4980 | 0.6512 |
| *Ru_qe*_50_70_5 (BM25L) | **0.5214** | **0.4250** | **0.3833** | 0.4974 | 0.6476 |

Table 6.6: Best teams results for Precision at 5 results for TREC PM 2017.

| Team | Run | Score |
|---|---|---|
| UD GU BioTM | UD_GU_CT_3 | 0.5500 |
| prna-mit-suny | pms_run5_tri | 0.4714 |
| udel | udelT2Comb | 0.4571 |
| NaCTeM | Broadc | 0.4571 |
| **NOVASearch** | NOVAtr2 | 0.4500 |
| UTDHLTRI | UTDHLTAFT | 0.4500 |
| UCAS | UCASSEM2 | 0.4429 |
| teckro | teckro1 | 0.4286 |
| CSIROmed | cCSIROmedAll | 0.4286 |
| KISTI | KISTI02CT | 0.4000 |

- if there is no way of improving these results with query expansion, the solution becomes to find better quality filters for excluding non relevant documents, even if they are related to the topic according to the text fields search.

### 6.2.6 NOVA at TREC PM 2017

The system developed on this work is an improved version of the NovaSearch system for TREC PM 2017 clinical trials track, where the information extraction techniques were used without much feedback for proper calibration, due to the lack of training data. The results of NovaSearch submissions were very good: one of the submitted runs was one of the two runs who was on Top 5 of the submitted runs for all the main metrics (P@5,P@10 and P@15), as shown on Tables 6.6, 6.7 and 6.8. These results were achieved by the run "NOVAtr2", which is similar to the run Ru_demo(BM25).

Table 6.7: Best team results for Precision at 10 results for TREC PM 2017.

| Team | Run | Score |
|---|---|---|
| UD GU BioTM | UD_GU_CT_3 | 0.4429 |
| UTDHLTRI | UTDHLTFFT | 0.4143 |
| teckro | teckro1 | 0.4000 |
| **NOVASearch** | NOVAtr2 | 0.4000 |
| udel | udelT2Comb | 0.3821 |
| UCAS | UCASSEM2 | 0.3786 |
| NaCTeM | Broadc | 0.3750 |
| KISTI | KISTI02CT | 0.3714 |
| POZNAN_SEMMED | LGDraw | 0.3714 |
| CSIROmed | cCSIROmedAll | 0.3679 |

Table 6.8: Best team results for Precision at 15 results for TREC PM 2017.

| Team | Run | Score |
|---|---|---|
| UD GU BioTM | UD_GU_CT_3 | 0.3881 |
| UTDHLTRI | UTDHLTAFT | 0.3762 |
| teckro | teckro1 | 0.3619 |
| UCAS | UCASSEM2 | 0.3548 |
| **NOVASearch** | NOVAtr2 | 0.3452 |
| POZNAN SEMMED | LGDraw | 0.3381 |
| udel | udelT2Comb | 0.3357 |
| KISTI | KISTI02CT | 0.3357 |
| CSIROmed | cCSIROmedAll | 0.3262 |
| prna-mit-suny | pms_run5_tri | 0.3262 |

## 6.3 Experiments and Results: supervised learning

### 6.3.1 Training and Test sets for supervised learning

The number of queries available on the ground truth for TREC PM 2017 was only 28 queries. Thus, the challenge was how to predict documents rankings for a given query without over-fitting the learned model to the query. In order to extract results in the most methodical fashion possible, "leave one out" cross validation was used which is explained on Section 6.3.2.

Twenty eight different training and test sets pairs were created, all with the same feature types. On each training/test pair set, one query was left out of the training set and it was used as the the only query on test set. This techniques improves the chances of getting a model that generalizes and returns unbiased results at testing time, as the only query that was not learned was the one being tested.

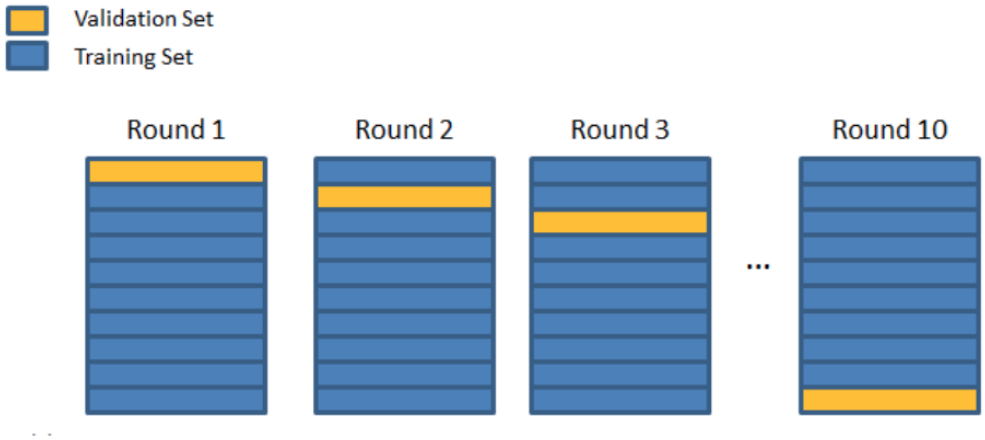


Figure 6.1: Example of cross validation training and validation data splits. Extracted from [9]

### 6.3.2 Cross Validation

Given the small number of queries presented to the learning system, and since we use ensemble methods with the occurrence of random values, cross validation was used to ensure that the results are stable, as described by [43], creating at learning time multiple validation sets from the initial training data, this way ensuring that each new fold created is somehow different from the previous folds. Figure 6.1 shows how each new fold (round) is created, each fold has exactly the same initial training data, but each validation and training set used at learning time has a small difference. On this case, five and ten folds cross validation was used, where each fold produces one model. The first CV implementation was with five folds and was used for parameter tuning. Ten folds were used to create the final ranking list so that we could have more data to train, reducing the number of data for validation. More folds (rounds) implies smaller number of queries for the validation set, because each query must be at least once represented on the validation set. Both cv implementations used leave one out cross validation where one query was initially extracted from the training set to be left out for testing, otherwise would be impossible to test all queries individually.

The first approach was to perform parameter tuning using multiple types of parameters combinations, creating five models for each different combination of parameters for a given algorithm. With five folds, four queries were used for validation, and on the last fold nine queries. After obtaining all results from the validations sets, we can calculate and average mean for each cross validation execution, and perceive which parameters are better suited for our problem. This first stage of cross validation was essential because the algorithms are designed for a wide range of problems, and without parameter tuning the parameters used may not be the best for that specific problem, leaving out possible better parameters combinations. This could lead to assuming a certain algorithm works poorly on the dataset, but instead is simply not tuned to it. Tables 6.10, 6.9 and 6.11 show the results of this parameter tuning for all three algorithms.

Table 6.9: Results for Cross validation on LambdaMART

| Parameter | Used Values | Best Value | Best value Avg. Precision |
|---|---|---|---|
| Learning Rate | 0.025/ 0.05/ **0.1**/ 0,2 | 0.1 | 0.1952 |
| Trees | 25/ 50/ 100/ 200/ 500/ **1000**/ 1500 | 1000 | 0.1949 |
| Leafs | 5/ **10**/ 40/ 50/ 100 | 10 | 0.2014 |

Table 6.10: Results for Cross validation on AdaRank

| Parameter | Used Values | Best Value | Best value Avg. Precision |
|---|---|---|---|
| Round | 100/ 200/ 700/ **500**/ 1000 | 500 | 0.2036 |
| Max | 2/ **5**/ 10/ 20/ 30 | 5 | 0.2105 |

Table 6.11: Results for Cross validation on Random Forests

| Parameter | Used Values | Best Value | Best value Avg. Precision |
|---|---|---|---|
| Bag | 600/ **300**/ 150/ 50 | 300 | 0.3435 |
| Trees | **1**/ 2/ 3 | 1 | 0.3411 |
| Leafs | 20/ 50/ **100**/ 150 | 100 | 0.3619 |
| Learning rate | 0.2/ 0.05/ **0.1** | 0.1 | 0,3410 |

After performing the parameter tuning process, the goal was the creation of a single ranking list. But since these algorithms performance change considerably with each new run, they could lead to either overfitted or underfitted models. So, cross validation was used, creating from each training set ten new validation sets (so ten new models for each query) using two queries from the training set to validate the models. Except for the last validation set which contain nine queries. RankLib developers [25] use this last model with more queries than in other models so that when extracting the results, developers can perform a controlled test to check the variance of the results of the learning system, so basically it checks the quality of the models from that run. If the variance is high between models created with two queries on validation set, and the ones created with more than two queries, then it shows that the models created may be over fitting the system. After creating ten models for each training set, we tested each individual model with the respective test set, and obtained ten ranking lists for each. So in order to obtain the final ranking for each query Reciprocal Rank Fusion was used, described on chapter 3, matching all ten ranking list and creating one single list.

### 6.3.3 Protocol

For the supervised learning implementation, it was necessary to transform the dataset into two distinct sets: A training set, to be used on the learning system, and a Test set, to be used on the ranking system. These sets were further subdivided into more training

and test sets, as explained above. The first one is created by extracting all documents existed on the index that are present on the relevance judgment provided by TREC PM 2017. This way, we can ensure that the learning system will have the maximum available information for him to correctly create a prediction/ranking model. Lucene was used to get all statistical and medical related information from the unsupervised learning previous implementation. This information is needed to construct the feature vector shown on 5.2. The Lucene index gives us the support to extract the necessary statistical information from each document-query pair (e.g., TF or IDF). After this step, the full dataset will be in following format:

Listing 6.2: Dataset format to perform LETOR

```
<line> .=. <target> qid:<qid> <feature>:<value>... <feature>:<value> # <info>
<target> : <positive integer>
<qid> : <positive integer>
<feature> : <positive integer>
<value> : <float>
<info> : <string>
```

where the feature id detailed on Table 5.2, is the positive integer on each <feature>, <value> is the value of the feature and it must be a continuous value, <qid> is the actual query number, <target> gives the relevance of that document for the query with the given id.

After the creation of the training set, the learning system can be executed to create a ranking model. Note that ten models will be created per execution since we are using 10-folds cross validation.

Since no groundtruth assumption should be made for the test set, a conventional IR ranking should be created, to obtain the documents that should be used for testing, extract statistical and medical information for them to use as a feature. In order to have the maximum correlation possible between the model and the ranking lists (test set), the documents that are considered relevant should be as fully represented as possible on the test set, which is achieved by using baselines for IR with high Recall results. Since we assume that we have no ground truth information when testing we did not use the baseline with the best recall, but the one that would be considered the best in terms of precision. which was still one of the best in terms of recall, so even doe is not the best we'll accept the lost of 10, 20 documents it won't retrieve. After the IR ranking is created the same idea as on the training set is used, using the document provided by the index to create their own feature vector, but on this case removing all queries except the one that is going to be used for testing.

Since this problem is going to be solved as an ordinal regression, each feature vector (document) is given a final score and they are not ordered by score, so all ranking list must be ordered, and later, RRF is used as explained on sub section 6.3.2 to obtain the final ranking list.

Table 6.12: Precision at 10 results for NovaSearch at TREC PM 2018.

| Run | Thesis implementation | Score (Median: 0.468) |
|---|---|---|
| Run 1 | *Ru_conditionrerank* | 0.544 |
| Run 2 | *Ru_manualexp* | 0.516 |
| Run 3 | *Rs_lambdaM* | 0.484 |
| Run 4 | *Rs_adarank* | 0.416 |

### 6.3.4 Features and Feature selection

Most of the created features were created based on both [42] and [40] LETOR implementations, were a lot of statistical information was extracted. Feature selection was fundamental on these types of features, these feature selection can be seen on Table A.5, showing each one of the feature ranked by their precision results when applying these features for retrieval. For example, three out of four LMJM features got low precision results on all four metrics (precision at 5, 10, 20, 30). For this reason all features with precision scores lower than the ones for feature 13 will be excluded. Gender and age specifications, were also removed because the retrieved documents had already passed a demographic filtering process, so these features were irrelevant.

### 6.3.5 NOVA at TREC PM 2018

Supervised learning was undoubtedly one of the main focus for NovaSearch team at TREC PM 2018. Leveraging on the ground truth assessments made for the previous year track, it was possible to utilize the 28 queries from 2017 dataset to train a LETOR model for each one of the used algorithms. Besides that, based on last year unsupervised learning implementations, new ideas for improving the retrieval were applied with very good results. Even doe both LambdaMart and Adarank runs were around the median and were still considered good, Table 6.12 clearly shows that the results of the unsupervised learning implementations improved the overall results of the system. Probably because of the lack of training data for both LETOR models.

To correctly understand the differences between both years implementations, all runs for the 2017 task were used on the 2018 dataset. What Figure 6.2 shows is that the supervised learning methods for 2018 were much more accurate then the ones for 2017 task (*Ru_demo* was the best from 2017 submission). Previous year best run results were around the same results as LambdaMart algorithm (run 3 of 2018 task), which was considered under fitted because of the lack of training data. So as this figure shows, from 2017 to 2018 the best run improved some very good 15%.

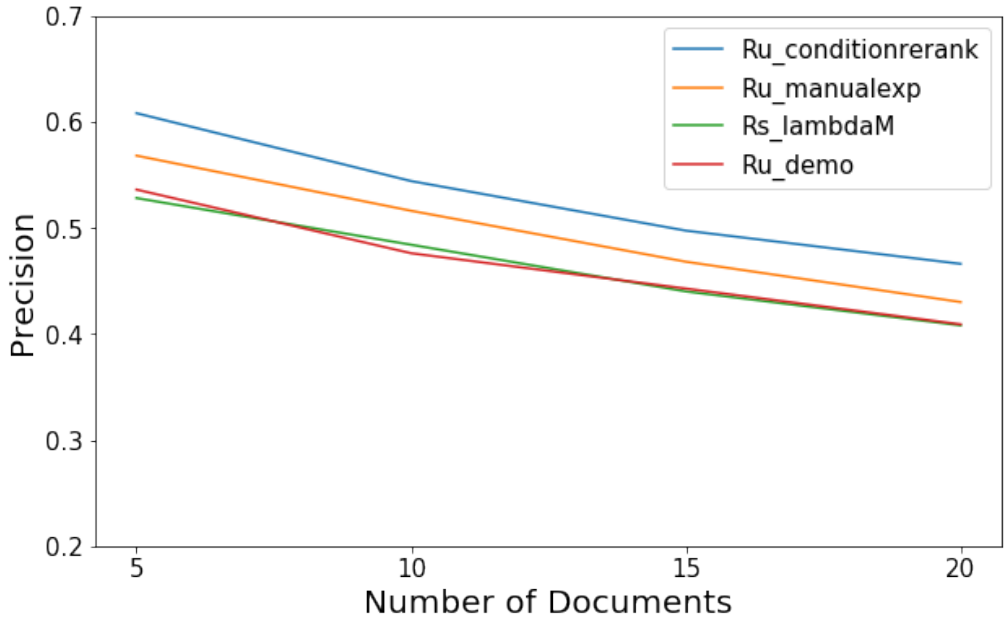


Figure 6.2: Precision at 5, 10 and 15 for the best run for 2017 and the best runs from 2018 task

## 6.4 Unsupervised learning Discussion

The overall results for the Clinical Trials retrieval were very positive when compared with the TREC PM 2017 results on [4]. We introduced methods such as rank fusion, filters based on the demographic information of the patient, boosting documents according to cancer related trials suppositions, or even re rank these documents according to some specific fields on each clinical trial. All these methods were rearranged to create multiple good solutions for the retrieval on the clinical trials dataset. After inspecting the relevance judgment file (i.e. file containing a list of files relevant and non relevant to the predefined queries) provided by TREC, one can infer that the problem is mainly with the exclusion criteria. The hypothesis is that is not being correctly used and so all query related documents will appear, even those that should be excluded for the given query.

By examining the results from the re-ranking implementations (the best results for this system), one can see that using the condition field can really improve results, but some of the old problems still persists. It is still very difficult for the system to be precise when dealing with queries that have got very small number of relevant documents (some with only 2 or 3 relevant documents in all clinical trials dataset). Figure 6.3 shows the benchmarks created for the unsupervised system, where each new benchmark represent new ideas and implementations, in a way that we can incrementally start from the benchmark with the worst performance until we reach the one with best performance by adding up new querying methods.

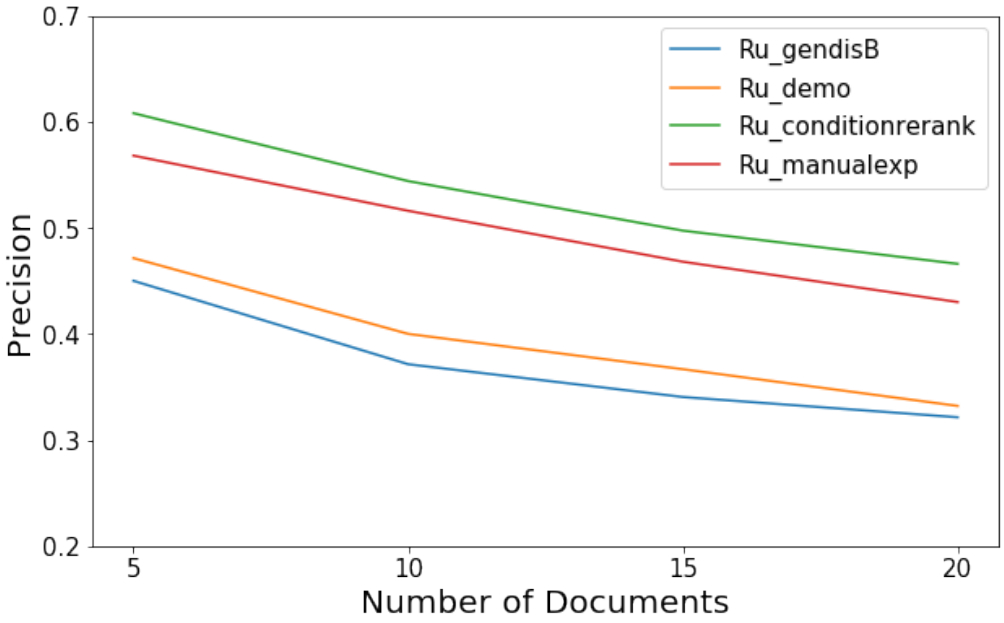


Figure 6.3: Precision at 10 for unsupervised learning benchmarks.

## 6.5 Supervised learning Discussion

The three used algorithms were considered feasible for real-world cases, obtaining acceptable results even on some of the already proven to be hard to answer queries. Despite that, results of unsupervised learning system (information retrieval) were overall better than the supervised ones, reducing precision results by a margin of about 10%, 20%. Given the limitations at hand, we can assume that these results will even out or improve if a real-world case with more available queries. In terms of individual results, Table 6.13 shows Random forests was the algorithm with the best results, followed by lambdaMART and AdaRank. The difference on the results between algorithms can definitely be explained by the type of learning to rank method used. For example, lambdaMART and random forests were both pairwise, while AdaRank is a listwise approach. Our assumption is that it needed more training data in order to correctly model the problem. As Table 6.13 shows and as it was expected, training these algorithms with more queries increases the overall quality of the system. Since TREC PM 2018 groundtruth files are already acessible for participants, all training set was created with the 50 queries for 2018 track, and in order to correctly compare the results the test set was composed by 2017 task queries. Always comparing to the system with best results, which still had better results than the new random forest system.

One very positive point extracted from the results (Random Forests and lambdaMART baselines) was the improvement in some of the more difficult queries. Query 5 is a perfect example of this, were the learning to rank algorithms used the combination of all features to understand which documents are probably more relevant, and got improvements of

Table 6.13: Results from learning to rank algorithms

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| *Rs_lambdaM* | 0.4571 | 0.3929 | 0.3452 | 0.5189 | 0.7175 |
| *Rs_randomF* | 0.4929 | 0.4179 | 0.3571 | 0.5321 | 0.7175 |
| *Rs_randomF* (50 queries train) | 0.5143 | 0.4357 | 0.3762 | 0.5382 | 0.7175 |
| *Rs_adarank* | 0.3357 | 0.2643 | 0.2238 | 0.4373 | 0.7175 |

almost 30% on Rs_randomF (random forest) and Rs_lambdaM (lambdaMART). To corroborate this, Figure 6.4 shows the individual query results for the system trained with 50 queries and the same individual results for the best unsupervised learning implementation *Ru_conditionrerank*. What is very relevant on these results is that it was the first system that had at least one relevant document on the top-5 (P@5>0) for all queries.

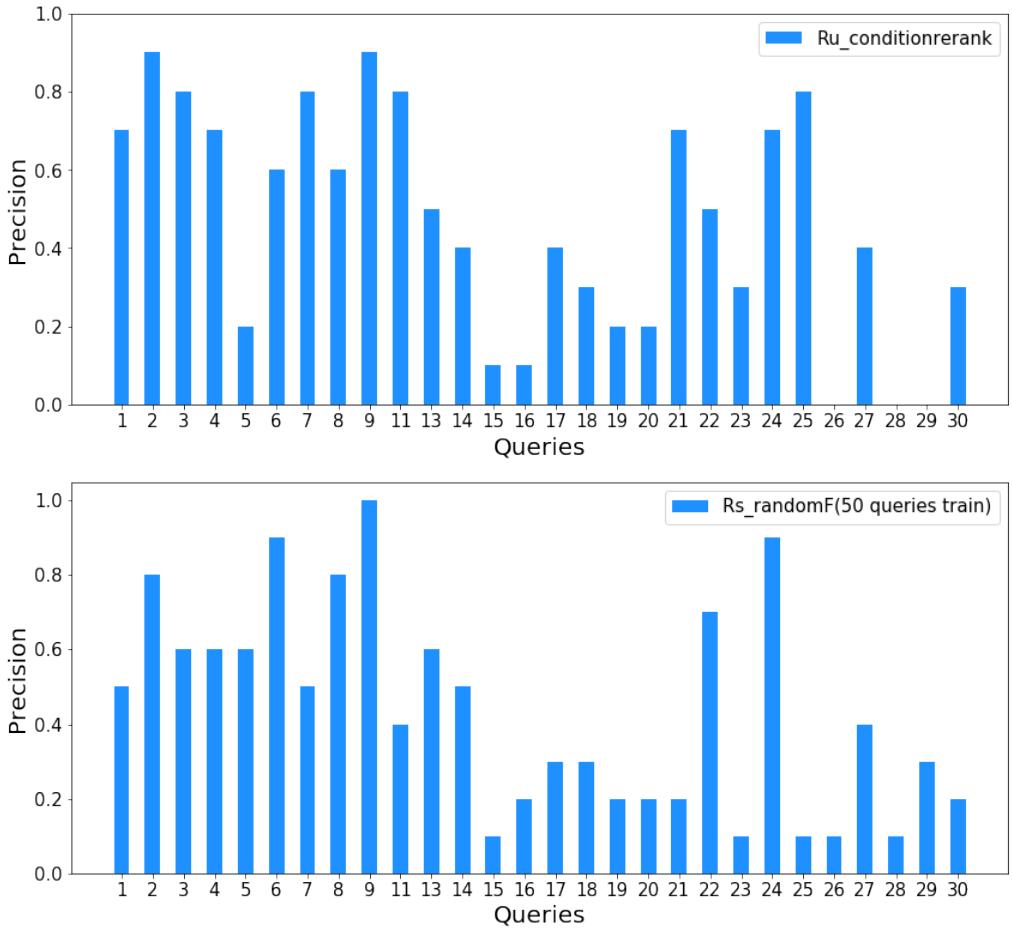


Figure 6.4: Each query P@10 results for compare *Ru_conditionrerank* and *Rs_randomF* (trained with 50 queries)

The analysis of these results leads us to believe if we had even more queries to train, the results could surpass the information retrieval system. improving not only top performing, but also the more difficult queries with very small number of assessed relevant documents

(queries 26, 28 are a good example). Supervised learning may be the solution for these types of problems in real life, capable of being applied unto an medical clinic or a hospital. In both scenarios, the amount of training data will increase with further editions of TREC PM, leading to an increase on the systems quality over time and not a static solution. This is key for an always mutating domain like oncology, where new discoveries are performed daily.

CHAPTER 7

# CONCLUSION

This thesis work shows two very different approaches to the same problem, matching patients to clinical trials, using unsupervised and supervised learning. Both approaches use information extraction to later perform information retrieval and/or supervised learning to better retrieve information. Even though the limitations of the TREC PM track (e.g. the small number of queries and examples) distantiate the work from real-world cases. The heterogeneity between patients information from clinical trials used on this work, give us some kind of assurance that those results will probably be acceptable for real-world cases, and work as good or better than it worked on this controlled environment.

## 7.1 Contributions

The contributions of this thesis are divided into two parts. The first one is the information extraction performed, which is capable of helping other researchers and developers understand how to deal with oncology domain problems and the types of information that can be extracted from medical related text and it's application to retrieval. For example, which information should be used to search in a more broad manner and which information should be used in a more fine-grained filtering (e.g., to automatically include or exclude documents).

The second part, is the application and thorough benchmarking of multiple information retrieval and machine learning algorithms to the information extracted. This thesis can help on understanding which methods achieve good results for this types of expert domain problems, both from a technical and domain related perspective.

## 7.2 Impact

This thesis work is a good example on how can computer science be used to enhance other areas, more specifically on this case oncology treatments. Physicians and other healthcare experts can use any of the developed systems to reduce human workload for choosing the best trial for a patient. An ideal system would have professionals need only to inspect top 10 retrieved documents, instead of having to methodically inspect dozens or hundreds of documents, in a very exhaustive and fault prone manner.

Both systems are a good fit for medical institutions where a lot of information flows in daily, both being able to dissect clinical trials and understand which ones are more relevant to the patient. The only difference between both system is that, the supervised learning implementation can be a more reliable solution to be applied in long term solutions, to aid physicians to continuously improve their system instead of using a static solution, that could only be upgraded by an developer.

## 7.3 Achievements

1. **TREC PM submission**: The first achievement was the participation on both TREC Precision Medicine 2017 and 2018 challenges, which provided this thesis with a use case very similar to the ones on real-world. Besides that, since it was a competition it was easy to understand where our ideas and systems stand, compared to a group of teams from all around the world. This greatly helps seeing which implementations were similar to other teams, and which ideas were completely new and effective to the problem.

2. **Methodically detailed work**: the goal of this work was to create a guidebook, to help other researchers and developers understand what methods can be used on other ranking problems, which implementations proved to be good and why. In addition, it also helps distinguishing between the good and bad implementations and try to overcome some persistent problems. All experiments and results were methodically implemented and later tested, always creating this work in an iterative manner, adding methods to the system layer by layer.

3. **Information retrieval explained to Medical personal and other domain experts**: This work shows to medical professionals limitations and difficulties developers have when creating expert systems. It shows what should be their concerns when writing clinical trials or other medical essays to improve the quality of such retrieval systems. For example, create textual fields without irrelevant and misleading information. Criteria field is a very good example of this, since it is without the doubt, a hard field to extract precise information from. Not that the medical literature is being incorrectly done, but if medical professionals are more aware of developers problems then both can benefit from it.

For researchers and developers, this work shows which information is truly indispensable in order to achieve good results when creating systems related to oncology. A lot of features and proven good implementations were made by performing small assumptions about clinical trials and cancer treatment. These ideas also apply to other problems in the oncology domain.

## 7.4 Limitations and Future Work

As it was explained throughout this document, the one true limitation on this work was the reduced number of queries (28+50) and ,consecutively small number of relevant documents to infer. This led to an increased difficulty for algorithms to perceive which features characterize relevant documents. In addition, limitations in terms of computation power, for instance, when extracting keywords it would be interesting to extract keywords from all documents at least one time, but given the amount of time it took to extract from one single document, it was expected that the full dataset would last at least three/four weeks to get all keywords.

For future work, I think is necessary to split it into three parts. The first one is improvement of some retrieval methods: utilize more sophisticated systems as Stanford PoS tagger [62] to extract more complex information, as exclusion criteria which was one of the big struggles to perform this work. This would enable understanding what else could be missing in terms of domain specific information or could be extracted from the genes information (for example understand lineage between genes, difference between a certain gene being the first to appear or being the second or third).

The second part of this future work should be focused on improving the features precision and create new features for learning to rank problems. Other possibility would be to try out different algorithms inside the decision trees, or experiment with other machine learning frameworks as neural networks.

The last should be try to implement this type of system in a real-world case, using a different dataset, to corroborate some of the decisions made during the systems implementation. This could be combined with a user friendly visualization tool, that helps physicians quickly use the system and enroll patients into clinical trials as soon as possible.

# BIBLIOGRAPHY


[1] S. R.P.M.C.W.S.M.K.V.-S. A. S. M. Ashique Mahmood Gang Li. "UD_GU_BioTM at TREC 2017: Precision Medicine Track". In: (2017).

[2] A. B. Abacha. "NLM NIH at TREC 2016 Clinical Decision Support Track." In: *TREC*. 2016.

[3] A. B. Abacha and S. Khelifi. "LIST at TREC 2015 Clinical Decision Support Track: Question Analysis and Unsupervised Result Fusion." In: *TREC*. 2015.

[4] G. Araújo, A. Mourao, and J. Magalhaes. "NOVASearch at Precision Medicine 2017". In: (2017).

[5] A. R. Aronson and T. C. Rindflesch. "Query expansion using the UMLS Metathesaurus." In: (1997), p. 485.

[6] J. Bai, D. Song, P. Bruza, J.-Y. Nie, and G. Cao. "Query expansion using term relationships in language models for information retrieval". In: *Proceedings of the 14th ACM international conference on Information and knowledge management*. ACM. 2005, pp. 688–695.

[7] N. J. Belkin and W. B. Croft. "Information filtering and information retrieval: Two sides of the same coin?" In: *Communications of the ACM* 35.12 (1992), pp. 29–38.

[8] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[9] S. J. blog. *Cross Validation and the Bias-Variance tradeoff (for Dummies)*. URL: https://codesachin.wordpress.com/2015/08/30/cross-validation-and-the-bias-variance-tradeoff-for-dummies/.

[10] L. Breiman. "Bagging predictors". In: *Machine learning* 24.2 (1996), pp. 123–140.

[11] L. Breiman. "Random forests". In: *Machine learning* 45.1 (2001), pp. 5–32.

[12] L. Breiman. *Classification and regression trees*. Routledge, 2017.

[13] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. "Learning to Rank using Gradient Descent". In: ().

[14] C. J. Burges. *From RankNet to LambdaRank to LambdaMART: An Overview*. Tech. rep. 2010. URL: https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/.

[15] S. Büttcher, C. L. Clarke, and G. V. Cormack. *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.

[16] G. Cao, J.-Y. Nie, J. Gao, and S. Robertson. "Selecting good expansion terms for pseudo-relevance feedback". In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2008, pp. 243–250.

[17] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. "Adapting ranking SVM to document retrieval". In: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2006, pp. 186–193.

[18] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. "Learning to rank: from pairwise approach to listwise approach". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 129–136.

[19] O. Chapelle and Y. Chang. "Yahoo! learning to rank challenge overview". In: *Proceedings of the Learning to Rank Challenge*. 2011, pp. 1–24.

[20] S. Choi and J. Choi. *SNUMedinfo at TREC CDS track 2014: Medical case-based retrieval task*. Tech. rep. Seoul National Univ (Republic of Korea), 2014.

[21] K. B. Cohen and L. Hunter. "Natural language processing and systems biology". In: *Artificial intelligence methods and tools for systems biology*. Springer, 2004, pp. 147–173.

[22] G. V. Cormack, C. L. A. Clarke, and S. Buettcher. "Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods". In: *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: ACM, 2009, pp. 758–759. ISBN: 978-1-60558-483-6. DOI: 10.1145/1571941.1572114. URL: http://doi.acm.org/10.1145/1571941.1572114.

[23] D. Cossock and T. Zhang. "Subset ranking using regression". In: *International Conference on Computational Learning Theory*. Springer. 2006, pp. 605–619.

[24] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. "A framework for selective query expansion". In: *Proceedings of the thirteenth ACM international conference on Information and knowledge management*. ACM. 2004, pp. 236–237.

[25] V. Dang. *Learning to rank implementation*. URL: https://sourceforge.net/p/lemur/wiki/RankLib/.

[26] T. G. Dietterich. "Ensemble Methods in Machine Learning". In: ().

[27] K. Donnelly. "SNOMED-CT: The advanced terminology and coding system for eHealth". In: *Studies in health technology and informatics* 121 (2006), p. 279.

[28] D. Frank Hsu and I. Taksa. "Comparing Rank and Score Combination Methods for Data Fusion in Information Retrieval". In: *Information Retrieval* 8.3 (2005), pp. 449–480. ISSN: 1573-7659. DOI: 10.1007/s10791-005-6994-4. URL: https://doi.org/10.1007/s10791-005-6994-4.

[29] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. "An efficient boosting algorithm for combining preferences". In: *Journal of machine learning research* 4.Nov (2003), pp. 933–969.

[30] Y. Freund, R. Schapire, and N. Abe. "A short introduction to boosting". In: *Journal-Japanese Society For Artificial Intelligence* 14.771-780 (1999), p. 1612.

[31] N. Fuhr. "Optimum polynomial retrieval functions based on the probability ranking principle". In: *ACM Transactions on Information Systems (TOIS)* 7.3 (1989), pp. 183–204.

[32] M. C. Genome. *Find my Cancer Genome*. URL: https://www.mycancergenome.org/content/disease/melanoma/nras/75/.

[33] T. R. Goodwin, M. A. Skinner, and S. M. Harabagiu. "UTD HLTRI at TREC 2017: Precision Medicine Track". In: (2017).

[34] A. García Seco de Herrera, J. Kalpathy-Cramer, D. Demner Fushman, S. Antani, and H. Müller. "Overview of the ImageCLEF 2013 medical tasks". In: *Working Notes of CLEF 2013 (Cross Language Evaluation Forum)*. Valencia, Spain, 2013.

[35] N. C. Institute. *Cancer Statistics*. URL: https://www.cancer.gov/about-cancer/understanding/statistics.

[36] A. E. W. Johnson, M. M. Ghassemi, S. Nemati, K. E. Niehaus, D. A. Clifton, and G. D. Clifford. "Machine Learning and Decision Support in Critical Care". In: *Proceedings of the IEEE* 104.2 (2016), pp. 444–466. ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2501978.

[37] W. A. Kibbe, C. Arze, V. Felix, E. Mitraka, E. Bolton, G. Fu, C. J. Mungall, J. X. Binder, J. Malone, D. Vasant, et al. "Disease Ontology 2015 update: an expanded and updated database of human diseases for linking biomedical knowledge through disease data". In: *Nucleic acids research* 43.D1 (2014), pp. D1071–D1078.

[38] M. Knowles and P. J. Selby. *Introduction to the cellular and molecular biology of cancer*. Oxford university press, 2005.

[39] A. Kumar. *Rank level fusion*. 2009.

[40] H. Li. "Learning to rank for information retrieval and natural language processing". In: *Synthesis Lectures on Human Language Technologies* 4.1 (2011), pp. 1–113.

[41] C. E. Lipscomb. "Medical subject headings (MeSH)". In: *Bulletin of the Medical Library Association* 88.3 (2000), p. 265.

[42] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. "Letor: Benchmark dataset for research on learning to rank for information retrieval". In: *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*. Vol. 310. ACM Amsterdam, The Netherlands. 2007.

[43] T.-Y. Liu et al. "Learning to rank for information retrieval". In: *Foundations and Trends® in Information Retrieval* 3.3 (2009), pp. 225–331.

[44] J. A. Lossio-Ventura, C. Jonquet, M. Roche, and M. Teisseire. "Biomedical term extraction: overview and a new methodology". In: *Information Retrieval Journal* 19.1-2 (2016), pp. 59–99.

[45] Y. Lv and C. Zhai. "When Documents Are Very Long, BM25 Fails!" In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. Beijing, China: ACM, 2011, pp. 1103–1104. ISBN: 978-1-4503-0757-4. DOI: 10.1145/2009916.2010070. URL: http://doi.acm.org/10.1145/2009916.2010070.

[46] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715.

[47] I. J. Marshall, J. Kuiper, and B. C. Wallace. "RobotReviewer: evaluation of a system for automatically assessing bias in clinical trials". In: *Journal of the American Medical Informatics Association* (2015), ocv044. DOI: 10.1093/jamia/ocv044. URL: http://dx.doi.org/10.1093/jamia/ocv044.

[48] M. McCandless, E. Hatcher, and O. Gospodnetic. *Lucene in Action: Covers Apache Lucene 3.0*. Manning Publications Co., 2010.

[49] A. Mourão, F. Martins, and J. Magalhães. "NovaSearch at TREC 2014 Clinical Decision Support Track". In: (). URL: http://trec.nist.gov/pubs/trec23/papers/pro-NovaSearch_clinical.pdf.

[50] A. Mourão, F. Martins, and J. Magalhães. "NovaSearch at TREC 2015 Clinical Decision Support Track". In: (). URL: http://trec.nist.gov/pubs/trec24/papers/NovaSearch-CL.pdf.

[51] A. Mourao, F. Martins, and J. Magalhaes. "NovaSearch on Medical ImageCLEF 2013." In: (2013).

[52] A. Mourão, F. Martins, and J. Magalhães. "Multimodal medical information retrieval with unsupervised rank fusion". In: *Computerized Medical Imaging and Graphics* 39 (2015), pp. 35–45.

[53] H.-S. Oh and Y. Jung. *KISTI at TREC 2014 Clinical Decision Support Track: Concept-based Document Re-ranking to Biomedical Information Retrieval*. Tech. rep. KOREA INST OF SCIENCE and TECHNOLOGY INFORMATION DAEJEON (KOREA), 2014.

[54] scikit-learn: machine learning in Python. *Decision Tree Regression*. URL: http://ogrisel.github.io/scikit-learn.org/sklearn-tutorial/auto_examples/tree/plot_tree_regression.

[55] J. R. Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (1986), pp. 81–106.

[56] K. Roberts, D. Fushman, E. Voorhees, W. Hersh, S. Bedrick, A. Lazar, and S. Pant. "Overview of the TREC 2017 Precision Medicine Track". In: (2017).

[57] S. Robertson and H. Zaragoza. "The Probabilistic Relevance Framework: BM25 and Beyond". In: *Foundations and Trends in Information Retrieval* 3.4 (2009), pp. 333–389.

[58] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. C. Kipper-Schuler, and C. G. Chute. "Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications". In: *Journal of the American Medical Informatics Association* 17.5 (2010), pp. 507–513.

[59] H. Scells, G. Zuccon, B. Koopman, A. Deacon, L. Azzopardi, and S. Geva. "Integrating the Framing of Clinical Questions via PICO into the Retrieval of Medical Literature for Systematic Reviews". In: (Nov. 2017), pp. 2291–2294.

[60] A. Stevens, J. S. Lowe, and I. Scott. *Core pathology*. Elsevier Health Sciences, 2008.

[61] T. Technology. *How to do a linear regression with sklearn*. URL: https://tutorials.technology/tutorials/19-how-to-do-a-regression-with-sklearn.

[62] K. Toutanova and C. D. Manning. "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-speech Tagger". In: *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora: Held in Conjunction with the 38th Annual Meeting of the Association for Computational Linguistics - Volume 13*. EMNLP '00. Hong Kong: Association for Computational Linguistics, 2000, pp. 63–70. DOI: 10.3115/1117794.1117802. URL: https://doi.org/10.3115/1117794.1117802.

[63] A. Trotman. "Learning to rank". In: *Information Retrieval* 8.3 (2005), pp. 359–381.

[64] Z.-G. Wang, L. Zhang, and W.-J. Zhao. "Progress in digital medicine". In: *Chinese Journal of Traumatology* 20.5 (2017), p. 297.

[65] *What is clinical research?* http://www.uhs.nhs.uk/ClinicalResearchin\Southampton/Public-and-patients/What-is-clinical-research.aspx.

[66] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao. "Adapting boosting for information retrieval measures". In: *Information Retrieval* 13.3 (2010), pp. 254–270.

[67] J. Xu and W. B. Croft. "Query expansion using local and global document analysis". In: *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 1996, pp. 4–11.

[68] J. Xu and H. Li. "Adarank: a boosting algorithm for information retrieval". In: *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2007, pp. 391–398.

[69] T. Xu, D. W. Oard, and P. McNamee. "HLTCOE at TREC 2014: Microblog and Clinical Decision Support." In: *TREC*. 2014.

APPENDIX A

# APPENDIX

## A.1 Rank Fusion

This technique is very common among IR system, and as proven to performed well in multiple systems. Two distinct rank fusion algorithms were implemented: Reciprocal Rank Fusion (RRF) and CombSUM. Both algorithm were implemented combining results from IR methods and based on section 3.2.4 formulas for each algorithm. The following Table A.1 shows the Rank Fusion runs, detailing the methods used on each one (all runs were tested with RRF and CombSum algorithms).

Table A.1: Rank fusion runs composition.

| Runs | Ranking 1 | Ranking 2 | Ranking 3 |
|---|---|---|---|
| RF1 | $Ru\_demo(BM25)$ | $Ru\_domainBoosted(BM25)$ | $Ru\_domainBoosted(LMD)$ |
| RF2 | $Ru\_demo(LMD)$ | $Ru\_domainBoosted(LMD)$ | $Ru\_domainBoosted(BM25)$ |
| RF3 | $Ru\_demo(BM25L)$ | $Ru\_domainBoosted(BM25L)$ | $Ru\_domainBoosted(BM25)$ |
| RF4 | $Ru_qe$_50_70_5 | $Ru\_domainBoosted(BM25)$ | $Ru_qe$_50_50_5 |
| RF5 | $Ru\_domainBoosted(BM25L)$ | $Ru\_domain(BM25L)$ | $Ru_qe$_50_50_5 |
| RF6 | $Ru\_domainBoosted(BM25L)$ | $Ru\_int(LMD)$ | $Ru\_demo(LMD)$ |

### A.1.1 Rank Fusion Experiments

As we can see by the results on Table A.2, both rank fusion algorithms used cam improve the results. Combining some of the previous techniques leading to good values in precision, Discount Cumulative Gain and Recall. The problem with this technique is to understand which ranking lists should be used for the fuse. A good example is baseline CombSum4 which is formed by: $Ru_qe$_50_70_5$(BM25L)$, $Ru\_domainBstd(BM25)$ and $Ru_qe$_50_50_5$(LMD)$. Although, the P@5 and P@10 values were the best among all other rank fusion baselines, the rankings involved on the fuse were not the ones with the biggest

Table A.2: Results for the rank fusion technique.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| CombSum 4 | **0.5214** | **0.4321** | 0.3786 | 0.4933 | 0.6484 |
| CombSum 5 | 0.5143 | 0.4214 | 0.3667 | 0.4965 | 0.6494 |
| CombSum 6 | 0.4929 | 0.4286 | 0.3762 | **0.5251** | **0.7167** |
| RRF 4 | 0.5000 | **0.4321** | 0.3786 | 0.4964 | 0.6475 |
| RRF 5 | **0.5214** | 0.4250 | 0.3738 | 0.5355 | **0.7557** |
| RRF 6 | 0.4857 | 0.4250 | 0.3810 | **0.5399** | 0.7551 |

Table A.3: Used ranking lists for rank fusion.

| Runs | Ranking 1 | Ranking 2 | Ranking 3 |
|---|---|---|---|
| RF4 | $Ru\_qe\_50\_70\_5$ | $Ru\_domainBoosted(BM25)$ | $Ru\_qe\_50\_50\_5$ |
| RF5 | $Ru\_domainBoosted(BM25L)$ | $Ru\_domain(BM25L)$ | $Ru\_qe\_50\_50\_5$ |
| RF6 | $Ru\_domainBoosted(BM25L)$ | $Ru\_int(LMD)$ | $Ru\_demo(LMD)$ |

precision results, Table A.3 shows each one of the fusions made for the results on Table A.2. To solve this problem, algorithms like the Learning to Fuse (L2F) or LETOR referred on Section 3.2.4 could be very useful to overcome this problem and further increase the results.

## A.2 Patients clinical records

Table A.4: Full list of clinical trials query topics: disease, gene, demographic, and other.

| Topic | Disease | Gene | Demographic | Other |
|---|---|---|---|---|
| 1 | Liposarcoma | CDK4 Amplification | 38-year-old male | GERD |
| 2 | Colon cancer | KRAS (G13D), BRAF (V600E) | 52-year-old male | Type II Diabetes, Hypertension |
| 3 | Meningioma | NF2 (K322), AKT1(E17K) | 45-year-old female | None |
| 4 | Breast cancer | FGFR1 Amplification, PTEN (Q171) | 67-year-old female | Depression, Hypertension, Heart Disease |
| 5 | Melanoma | BRAF (V600E), CDKN2A Deletion | 45-year-old female | None |
| 6 | Melanoma | NRAS (Q61K) | 55-year-old male | Hypertension |
| 7 | Lung cancer | EGFR (L858R) | 50-year-old female | Lupus |
| 8 | Lung cancer | EML4-ALK Fusion transcript | 52-year-old male | Hypertension, Osteoarthritis |
| 9 | Gastrointestinal stromal tumor | KIT Exon 9 (A502_Y503dup) | 49-year-old female | None |
| 11 | Gastric cancer | PIK3CA (E545K) | 54-year-old male | Depression |
| 13 | Cholangiocarcinoma | BRCA2 | 72-year-old male | Diabetes |
| 14 | Cholangiocarcinoma | IDH1 (R132H) | 64-year-old male | Neuropathy |
| 15 | Cervical cancer | STK11 | 26-year-old female | None |
| 16 | Pancreatic cancer | CDKN2A | 54-year-old male | Diabetes, Hypertension |
| 17 | Prostate cancer | PTEN Inactivating | 81-year-old male | Hypertension, Depression |
| 18 | Pancreatic cancer | CDK6 Amplification | 48-year-old male | None |
| 19 | Colorectal cancer | FGFR1 Amplification | 35-year-old female | None |
| 20 | Liposarcoma | MDM2 Amplification | 26-year-old male | None |
| 21 | Lung adenocarcinoma | ALK Fusion | 64-year-old female | Emphysema |
| 22 | Lung cancer | ERBB2 Amplification | 70-year-old male | Arthritis |
| 23 | Breast cancer | PTEN Loss | 54-year-old female | Congestive Heart Failure |
| 24 | Lung cancer | NTRK1 | 58-year-old female | Depression, Hypertension, Diabetes |
| 25 | Lung adenocarcinoma | MET Amplification | 48-year-old male | Emphysema |
| 26 | Breast cancer | NRAS Amplification | 35-year-old female | None |
| 27 | Pancreatic adenocarcinoma | KRAS, TP53 | 49-year-old female | None |
| 28 | Pancreatic ductal adenocarcinoma | ERBB3 | 73-year-old female | Whipple, FNA |
| 29 | Ampullary carcinoma | KRAS | 61-year-old male | None |
| 30 | Pancreatic adenocarcinoma | RB1, TP53, KRAS | 57-year-old female | None |

## A.3 Feature selection

Table A.5: All features and respective P@5, 10, 20 and 30 for supervised learning. Intended for feature selection.

| Feature Number | P@5 | P@10 | P@20 | P@30 |
|---|---|---|---|---|
| 43 | 0.33571428 | 0.29642856 | 0.225 | 0.18214285 |
| 33 | 0.29285714 | 0.28214285 | 0.21607143 | 0.20833333 |
| 47 | 0.27857143 | 0.225 | 0.17321429 | 0.14404762 |
| 44 | 0.27142859 | 0.22857143 | 0.16964285 | 0.15357143 |
| 9 | 0.2642857 | 0.20714286 | 0.21428572 | 0.20595238 |
| 7 | 0.24285714 | 0.23928571 | 0.22321428 | 0.20119047 |
| 11 | 0.23571429 | 0.24285714 | 0.23392858 | 0.21309523 |
| 34 | 0.23571429 | 0.175 | 0.16428572 | 0.15357143 |
| 35 | 0.22142857 | 0.2 | 0.16607143 | 0.15476191 |
| 36 | 0.22142857 | 0.19642857 | 0.17142858 | 0.16547619 |
| 8 | 0.21428572 | 0.225 | 0.20535715 | 0.19285715 |
| 41 | 0.21428572 | 0.19285715 | 0.17678571 | 0.15952381 |
| 12 | 0.20714286 | 0.225 | 0.20535715 | 0.20952381 |
| 42 | 0.20714286 | 0.14642857 | 0.14642857 | 0.13452381 |
| 58 | 0.18571429 | 0.18928571 | 0.16964285 | 0.15119047 |
| 45 | 0.17857143 | 0.15 | 0.12321428 | 0.11309524 |
| 10 | 0.17142858 | 0.17142858 | 0.16428572 | 0.16309524 |
| 48 | 0.16428572 | 0.13571429 | 0.12142857 | 0.11071429 |
| 51 | 0.15714286 | 0.17857143 | 0.16785714 | 0.16071428 |
| 49 | 0.15 | 0.13214286 | 0.10892857 | 0.09166667 |
| 17 | 0.14285715 | 0.11071429 | 0.0875 | 0.08452381 |
| 20 | 0.13571429 | 0.09642857 | 0.08571429 | 0.08095238 |
| 57 | 0.12857144 | 0.11071429 | 0.09821428 | 0.0952381 |
| 46 | 0.12857144 | 0.10357143 | 0.094642855 | 0.098809525 |
| 19 | 0.12857144 | 0.08928572 | 0.08035714 | 0.08095238 |
| 52 | 0.114285715 | 0.08928572 | 0.07857143 | 0.07857143 |
| 18 | 0.114285715 | 0.075 | 0.071428575 | 0.07261905 |
| 54 | 0.10714286 | 0.1 | 0.08214286 | 0.06428572 |
| 15 | 0.10714286 | 0.075 | 0.0625 | 0.071428575 |
| 22 | 0.10714286 | 0.071428575 | 0.08571429 | 0.09166667 |
| 26 | 0.10714286 | 0.071428575 | 0.08571429 | 0.09166667 |
| 30 | 0.10714286 | 0.071428575 | 0.08571429 | 0.09166667 |
| 24 | 0.1 | 0.075 | 0.08571429 | 0.09166667 |
| 28 | 0.1 | 0.075 | 0.08571429 | 0.09166667 |
| 32 | 0.1 | 0.075 | 0.08571429 | 0.09166667 |

Table A.5 – continued from previous page

| Feature Number | P@5 | P@10 | P@20 | P@30 |
|---|---|---|---|---|
| 53 | 0.092857145 | 0.08928572 | 0.09821428 | 0.08571429 |
| 56 | 0.092857145 | 0.08928572 | 0.09821428 | 0.08571429 |
| 55 | 0.08571429 | 0.092857145 | 0.094642855 | 0.08452381 |
| 38 | 0.08571429 | 0.07857143 | 0.092857145 | 0.08571429 |
| 16 | 0.07857143 | 0.08214286 | 0.0875 | 0.083333336 |
| 21 | 0.071428575 | 0.08214286 | 0.08392857 | 0.092857145 |
| 25 | 0.071428575 | 0.08214286 | 0.08392857 | 0.092857145 |
| 29 | 0.071428575 | 0.08214286 | 0.08392857 | 0.092857145 |
| 13 | 0.06428572 | 0.071428575 | 0.073214285 | 0.065476194 |
| 14 | 0.05 | 0.060714286 | 0.051785715 | 0.046428572 |
| 39 | 0.042857144 | 0.05 | 0.042857144 | 0.04404762 |
| 40 | 0.035714287 | 0.05357143 | 0.0625 | 0.06904762 |
| 37 | 0.035714287 | 0.042857144 | 0.0625 | 0.06904762 |
| 23 | 0.035714287 | 0.03214286 | 0.046428572 | 0.046428572 |
| 27 | 0.035714287 | 0.03214286 | 0.046428572 | 0.046428572 |
| 31 | 0.035714287 | 0.03214286 | 0.046428572 | 0.046428572 |
| 50 | 0.028571429 | 0.017857144 | 0.04464286 | 0.05 |

## A.4 Results full extension

Table A.6: Demographic filter results.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| *Ru_gendisB* (BM25L) | 0.4500 | 0.3714 | 0.3405 | 0.5375 | 0.7563 |
| *Ru_age* (BM25L) | 0.4643 | 0.3786 | 0.3524 | 0.5416 | **0.7596** |
| *Ru_gender* (BM25L) | 0.4643 | 0.3929 | 0.3524 | 0.5426 | 0.7563 |
| *Ru_demo* (BM25L) | **0.4714** | **0.4000** | **0.3667** | **0.5455** | 0.7588 |
| *Ru_gendisB* (BM25) | 0.4286 | 0.3679 | 0.3405 | 0.5297 | 0.7496 |
| *Ru_age* (BM25) | 0.4429 | 0.3750 | 0.3571 | 0.5364 | 0.7534 |
| *Ru_gender* (BM25) | 0.4429 | 0.3929 | 0.3524 | 0.5368 | 0.7497 |
| *Ru_demo* (BM25) | 0.4500 | 0.3964 | 0.3643 | 0.5403 | 0.7528 |
| *Ru_gendisB* (LMD) | 0.4143 | 0.3857 | 0.3214 | 0.5067 | 0.7470 |
| *Ru_age* (LMD) | 0.4214 | 0.3857 | 0.3238 | 0.5105 | 0.7485 |
| *Ru_gender* (LMD) | 0.4286 | 0.4000 | 0.3286 | 0.5128 | 0.7471 |
| *Ru_demo* (LMD) | 0.4286 | **0.4036** | 0.3333 | 0.5154 | 0.7464 |
| *Ru_gendisB* (TFIDF) | 0.3643 | 0.3357 | 0.3000 | 0.4886 | 0.7083 |
| *Ru_age* (TFIDF) | 0.3500 | 0.3250 | 0.2905 | 0.4632 | 0.6945 |
| *Ru_gender* (TFIDF) | 0.3714 | 0.3429 | 0.3000 | 0.4747 | 0.6964 |
| *Ru_demo* (TFIDF) | 0.3714 | 0.3321 | 0.2929 | 0.4615 | 0.6937 |

Table A.7: Exclusion criteria results.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| *Ru_demo* (BM25L) | **0.4714** | 0.4000 | 0.3667 | 0.5455 | **0.7588** |
| *Ru_exclusion* (BM25L) | 0.4343 | 0.3964 | 0.3643 | 0.5165 | 0.7047 |
| *Ru_inclusion* (BM25L) | 0.4657 | **0.4107** | **0.3714** | **0.5500** | **0.8003** |
| *Ru_criteria* (BM25L) | 0.4143 | 0.3964 | 0.3452 | 0.4721 | 0.6668 |
| *Ru_demo* (BM25) | 0.4500 | 0.3964 | 0.3643 | 0.5403 | 0.7528 |
| *Ru_exclusion* (BM25) | 0.4229 | 0.3929 | 0.3643 | 0.5120 | 0.7002 |
| *Ru_inclusion* (BM25) | 0.4414 | 0.3964 | 0.3690 | 0.5511 | 0.7824 |
| *Ru_criteria* (BM25) | 0.4214 | 0.3821 | 0.3315 | 0.4684 | 0.6501 |
| *Ru_demo* (LMD) | 0.4286 | 0.4036 | 0.3333 | 0.5154 | 0.7464 |
| *Ru_exclusion* (LMD) | 0.4357 | 0.3893 | 0.3524 | 0.5544 | 0.7908 |
| *Ru_inclusion* (LMD) | 0.4286 | 0.3857 | 0.3333 | 0.4893 | 0.6879 |
| *Ru_criteria* (LMD) | 0.4071 | 0.3571 | 0.3238 | 0.4582 | 0.6541 |
| *Ru_demo* (TFIDF) | 0.3714 | 0.3321 | 0.2929 | 0.4615 | 0.6937 |
| *Ru_exclusion* (TFIDF) | 0.3714 | 0.3357 | 0.2905 | 0.4435 | 0.6604 |

Table A.8: Oncology filter results.

| **Baseline** | **P@5** | **P@10** | **P@15** | **NDCG** | **Recall** |
|---|---|---|---|---|---|
| *Ru_demo* (BM25L) | 0.4714 | 0.4000 | 0.3667 | **0.5455** | **0.7588** |
| *Ru_int* (BM25L) | 0.5000 | 0.4250 | 0.3857 | 0.5427 | 0.7389 |
| *Ru_pp* (BM25L) | 0.4857 | 0.4250 | 0.3738 | 0.5408 | 0.7357 |
| *Ru_st* (BM25L) | 0.4786 | 0.4179 | 0.3714 | 0.5407 | 0.7391 |
| *Ru_domain* (BM25L) | 0.5214 | 0.4250 | 0.3762 | 0.5108 | 0.6864 |
| *Ru_domainBoosted* (BM25L) | 0.5214 | 0.4179 | 0.3714 | 0.4938 | 0.6443 |
| *Ru_manualexp* (BM25L) | **0.5500** | **0.4500** | **0.3976** | 0.5243 | 0.7175 |
| *Ru_demo* (BM25) | 0.4500 | 0.3964 | 0.3643 | 0.5403 | 0.7528 |
| *Ru_int* (BM25) | 0.4786 | 0.4143 | 0.3690 | 0.5337 | 0.7383 |
| *Ru_pp* (BM25) | 0.4643 | 0.4107 | 0.3643 | 0.5273 | 0.7225 |
| *Ru_st* (BM25) | 0.4571 | 0.4179 | 0.3571 | 0.5280 | 0.7289 |
| *Ru_domain* (BM25) | 0.4929 | 0.4143 | 0.3762 | 0.4978 | 0.6633 |
| *Ru_domainBoosted* (BM25) | 0.5071 | 0.4143 | 0.3857 | 0.4855 | 0.6339 |
| *Ru_manualexp* (BM25) | 0.5286 | **0.4500** | 0.3786 | 0.5104 | 0.6826 |
| *Ru_demo* (LMD) | 0.4286 | 0.4036 | 0.3333 | **0.5154** | **0.7464** |
| *Ru_int* (LMD) | 0.5000 | 0.3929 | 0.3500 | 0.5085 | 0.7126 |
| *Ru_pp* (LMD) | 0.4714 | 0.3679 | 0.3286 | 0.4936 | 0.6908 |
| *Ru_st* (LMD) | 0.4714 | 0.3714 | 0.3262 | 0.4961 | 0.7021 |
| *Ru_domain* (LMD) | 0.5000 | 0.3857 | 0.3452 | 0.4715 | 0.6294 |
| *Ru_domainBoosted* (LMD) | 0.5000 | 0.3857 | 0.3452 | 0.4677 | 0.6220 |
| *Ru_manualexp* (LMD) | 0.5286 | **0.4500** | 0.3714 | 0.4996 | 0.6631 |
| *Ru_demo* (TFIDF) | 0.3714 | 0.3321 | 0.2929 | 0.4615 | 0.6937 |
| *Ru_int* (TFIDF) | 0.4286 | 0.3536 | 0.3048 | 0.4613 | 0.6647 |
| *Ru_pp* (TFIDF) | 0.4071 | 0.3357 | 0.2929 | 0.4494 | 0.6413 |
| *Ru_st* (TFIDF) | 0.4071 | 0.3393 | 0.3000 | 0.4543 | 0.6515 |
| *Ru_domain* (TFIDF) | 0.4429 | 0.3500 | 0.3119 | 0.4464 | 0.6047 |
| *Ru_domainBoosted* (TFIDF) | 0.4500 | 0.3714 | 0.3262 | 0.4479 | 0.5954 |

Table A.9: Results for the query expansion technique.

| **Baseline** | **P@5** | **P@10** | **P@15** | **NDCG** | **Recall** |
|---|---|---|---|---|---|
| *Ru_domain* (BM25L) | **0.5214** | **0.4250** | 0.3762 | **0.5108** | **0.6864** |
| *Ru_domainBoosted* (BM25L) | **0.5214** | 0.4179 | 0.3714 | 0.4938 | 0.6443 |
| *Ru_qe*_10_10_10 (BM25L) | **0.5214** | 0.3750 | 0.3119 | 0.4780 | 0.6337 |
| *Ru_qe*_10_10_5 (BM25L) | 0.5143 | 0.3857 | 0.3405 | 0.4881 | 0.6410 |
| *Ru_qe*_25_10_5 (BM25L) | 0.5143 | 0.3929 | 0.3405 | 0.4864 | 0.6410 |
| *Ru_qe*_25_25_5 (BM25L) | 0.4714 | 0.4036 | 0.3595 | 0.4904 | 0.6461 |
| *Ru_qe*_50_10_5 (BM25L) | 0.5143 | 0.3929 | 0.3405 | 0.4803 | 0.6401 |
| *Ru_qe*_50_25_5 (BM25L) | 0.4714 | 0.4071 | 0.3595 | 0.4898 | 0.6454 |
| *Ru_qe*_50_50_5 (BM25L) | **0.5214** | 0.4179 | 0.3738 | 0.4980 | 0.6512 |
| *Ru_qe*_50_70_5 (BM25L) | **0.5214** | **0.4250** | 0.3833 | 0.4974 | 0.6476 |
| *Ru_domainBoosted* (BM25) | 0.5071 | 0.4143 | **0.3857** | 0.4855 | 0.6339 |
| *Ru_qe*_10_10_10 (BM25) | 0.4857 | 0.3857 | 0.3333 | 0.4820 | 0.6352 |
| *Ru_qe*_10_10_5 (BM25) | 0.5000 | 0.4036 | 0.3476 | 0.4853 | 0.6389 |
| *Ru_qe*_25_10_5 (BM25) | 0.5143 | 0.4000 | 0.3476 | 0.4845 | 0.6389 |
| *Ru_qe*_25_25_5 (BM25) | 0.4714 | 0.4107 | 0.3762 | 0.4901 | 0.6425 |
| *Ru_qe*_50_10_5 (BM25) | 0.5143 | 0.3964 | 0.3500 | 0.4781 | 0.6373 |
| *Ru_qe*_50_25_5 (BM25) | 0.4714 | 0.4107 | 0.3690 | 0.4899 | 0.6430 |
| *Ru_qe*_50_50_5 (BM25) | 0.5071 | 0.4107 | 0.3786 | 0.4911 | 0.6406 |
| *Ru_qe*_50_70_5 (BM25) | 0.5000 | 0.4143 | **0.3857** | 0.4874 | 0.6360 |
| *Ru_domainBoosted* (LMD) | 0.5000 | 0.3857 | 0.3452 | 0.4677 | 0.6220 |
| *Ru_qe*_10_10_10 (LMD) | 0.4571 | 0.3321 | 0.2833 | 0.4345 | 0.6206 |
| *Ru_qe*_10_10_5 (LMD) | 0.4929 | 0.3679 | 0.3048 | 0.4513 | 0.6328 |
| *Ru_qe*_25_10_5 (LMD) | 0.4929 | 0.3679 | 0.3048 | 0.4513 | 0.6329 |
| *Ru_qe*_25_25_5 (LMD) | 0.4786 | 0.3821 | 0.3286 | 0.4604 | 0.6258 |
| *Ru_qe*_50_10_5 (LMD) | 0.5000 | 0.3643 | 0.3048 | 0.4485 | 0.6307 |
| *Ru_qe*_50_25_5 (LMD) | 0.4786 | 0.3857 | 0.3310 | 0.4598 | 0.6251 |
| *Ru_qe*_50_50_5 (LMD) | 0.5143 | 0.4071 | 0.3429 | 0.4701 | 0.6269 |
| *Ru_qe*_50_70_5 (LMD) | 0.5071 | 0.4000 | **0.3500** | **0.4706** | 0.6264 |
| *Ru_domainBoosted* (TFIDF) | 0.4500 | 0.3714 | 0.3262 | 0.4479 | 0.5954 |
| *Ru_qe*_10_10_10 (TFIDF) | 0.4429 | 0.3571 | 0.2952 | 0.4333 | 0.6056 |
| *Ru_qe*_10_10_5 (TFIDF) | 0.4714 | 0.3607 | 0.3071 | 0.4329 | 0.6054 |
| *Ru_qe*_25_10_5 (TFIDF) | 0.4714 | 0.3607 | 0.3071 | 0.4329 | 0.6054 |
| *Ru_qe*_25_25_5 (TFIDF) | 0.4714 | 0.3679 | 0.3119 | 0.4424 | 0.5900 |
| *Ru_qe*_50_10_5 (TFIDF) | 0.4643 | 0.3607 | 0.3048 | 0.4305 | 0.6042 |
| *Ru_qe*_50_25_5 (TFIDF) | 0.4714 | 0.3643 | 0.3119 | 0.4418 | 0.5888 |
| *Ru_qe*_50_50_5 (TFIDF) | 0.4714 | 0.3786 | 0.3262 | 0.4480 | 0.5967 |
| *Ru_qe*_50_70_5 (TFIDF) | 0.4571 | 0.3750 | 0.3333 | 0.4488 | 0.5969 |

Table A.10: Results for the rank fusion technique.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| CombSum 1 | 0.5071 | 0.4214 | **0.3857** | 0.4856 | 0.6349 |
| CombSum 2 | 0.5071 | 0.4179 | 0.3786 | 0.4874 | 0.6339 |
| CombSum 3 | 0.5143 | 0.4214 | 0.3690 | 0.4927 | 0.6443 |
| CombSum 4 | **0.5214** | **0.4321** | 0.3786 | 0.4933 | 0.6484 |
| CombSum 5 | 0.5143 | 0.4214 | 0.3667 | 0.4965 | 0.6494 |
| CombSum 6 | 0.4929 | 0.4286 | 0.3762 | **0.5251** | **0.7167** |
| RRF 1 | 0.5071 | 0.4250 | **0.3833** | 0.5340 | 0.7547 |
| RRF 2 | 0.4857 | 0.4214 | 0.3690 | 0.5313 | 0.7516 |
| RRF 3 | 0.5071 | 0.4214 | 0.3738 | 0.5325 | 0.7535 |
| RRF 4 | 0.5000 | **0.4321** | 0.3786 | 0.4964 | 0.6475 |
| RRF 5 | **0.5214** | 0.4250 | 0.3738 | 0.5355 | **0.7557** |
| RRF 6 | 0.4857 | 0.4250 | 0.3810 | **0.5399** | 0.7551 |

Table A.11: Results for the re ranking implementation.

| Baseline | P@5 | P@10 | P@15 | NDCG | Recall |
|---|---|---|---|---|---|
| $Ru\_excluderer rank(BM25L)$ | 0.5571 | 0.4357 | 0.3905 | 0.5220 | 0.7175 |
| $Ru\_conditionrerank(BM25L)$ | **0.5643** | **0.4607** | **0.4119** | **0.5331** | **0.7175** |
| $Ru\_rerank(BM25L)$ | 0.5571 | 0.4500 | **0.4119** | **0.5301** | **0.7175** |