**Project Idea Document**

CareFinder - Group 17

# **1. Abstract**

The US health care system is an enormous industry that is quite intransparent, making it fairly difficult to find the specialized health care one might be looking for. With CareFinder, we provide a solution. Users can search for doctors in the area, which are sorted on expertize rank. To give useful information based on a user search query, a couple of challenges are tackled using a series of information retrieval techniques. A set of medical specializations and keywords are retrieved by scraping and processing the content of multiple websites like Wikipedia through DBpedia. Doctors and the hospitals they are affiliated with, are retrieved by crawling hospital websites. To acquire missing hospital URL’s, crowdsourcing is employed. The expertize of a doctor or researcher is based on abstracts of papers and presence in clinical trials. Moreover, a model for finding doctor’s specialization is proposed. Search query results are ranked based on the user’s needs expressed in an expertise in a specialization. In order to realize the product, we have defined requirements, a project plan and product evaluation measures with regard to the performance of the system.

# **2. Introduction**

Health is by far the most important factor in life. On average, a US citizen spends over $9,000 on healthcare, beating every other country in the world[[1]](#footnote-0). With 5,500 hospitals, almost a million practicing doctors[[2]](#footnote-1) and 238,000 clinical trials[[3]](#footnote-2), finding the specialist care you are looking for is truly searching for a needle in a haystack. What makes this domain even more intransparent is the professional terminology which is a tough cookie for the average American. As a result, people spend days on finding the right medical specialist or results of the latest research to assure themselves of getting the best, state of the art treatment. But finding the perfect suitable care and evaluating doctors on their expertise is almost impossible.

CareFinder is the new ultimate aid for people seeking specialist care. This tool gives you the opportunity to search for doctors and clinical trials in your area and provides you with an expertise measurement based on (ongoing) clinical trials and published papers. Finding the right person or institute that can help you was never this easy and transparent.

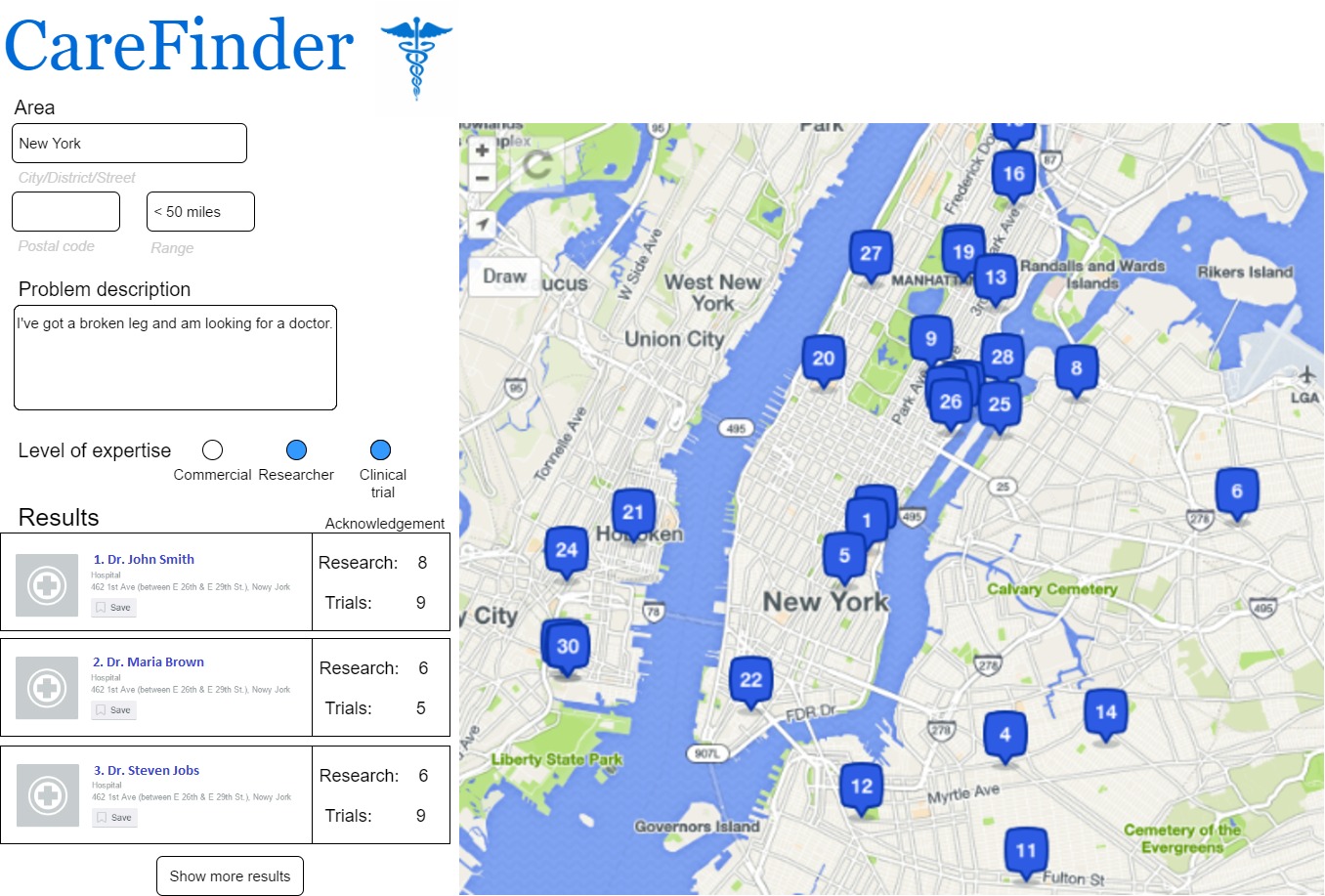
# **3. CareFinder, an IR system**

In this section we will introduce our solution to the general reader and yet use a minimum amount of scientific language. First we present an overview of the vision we have for CareFinder, the end we keep in mind[[4]](#footnote-3) while creating our application. We also discuss how the system would actually perform its job in an easy to follow manner. We furthermore discuss the innovations and challenges of our project and briefly discuss how this project relates to the material of the Information Retrieval course at Delft University of Technology.

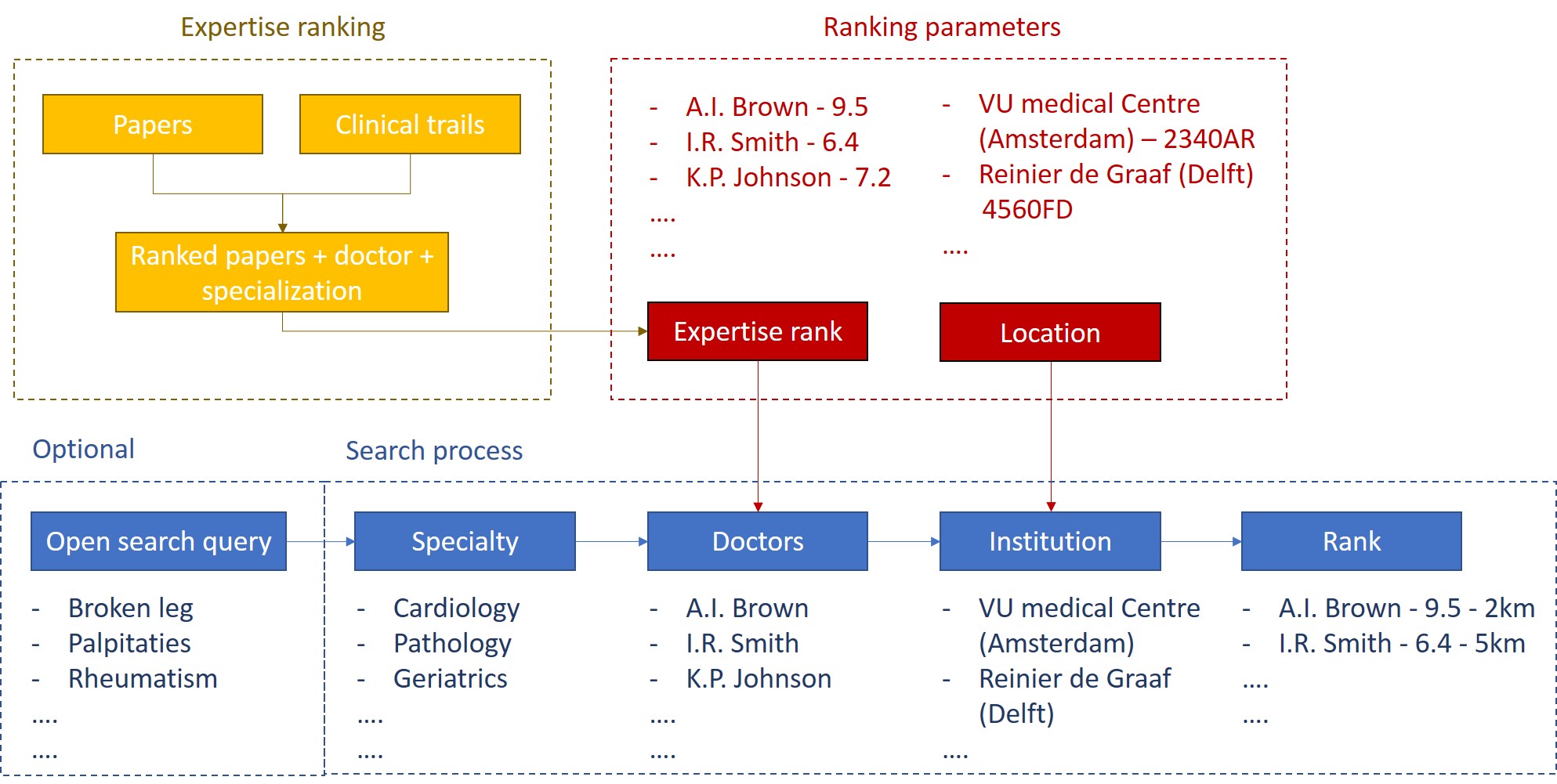
## **3.1 Overview**

In order for CareFinder to find you the best suitable care, you define a geographical search area based on your location and a range. For instance, all doctors and institutes within 50 miles of New York City. Additionally you describe your problem or the care you are looking for, for example “broken leg” or “orthopedist”. This open search query is an attractive feature but is not essential for the product. Alternatively search is done via selecting a specialization from a drop down menu. CareFinder also gives you the opportunity to specifically search for researchers, commercial doctors or clinical trials.

As output, CareFinder gives you a map of doctors and institutes that offer care in the field of your interest within the predefined location range. Furthermore it states all doctors with their function title and an expertise ranking based on research they performed in the past. These results can be sorted on expertise rank or location.



*Figure 1 - User interface of CareFinder showing results to a given query*



*Figure 2 - Block scheme of back end search process*

When a search query is given, CareFinder searches in its database for doctors within the medical specialisation specified in the query. The database lists all doctors. the hospital or institute they work at (of which the locations are retrieved) and their expertise grade based on scientific papers and clinical trials. Next, all doctors within the specified location range are filtered and sorted on location or expertise rank.

## **3.2 Innovations and challenges**

The section presents the innovation brought by CareFinder to the industry as well as the challenges that we have to face to develop and evaluate it.

### Innovative solution

There are various websites allowing finding the healthcare nearby, some even present doctors and their specialization field[[5]](#footnote-4). However, there is no platform allowing to easily find a doctor nearby, with the specialization that you need that also takes into account the scientific productivity and activity in clinical trials. Our system first of all provides multiple options for the search for instance**,** closest healthcare to a given address, query by specialization, by the free text describing what is wrong and also suitable fields to fill in case that you would like to contact doctor with some research or clinical trials background. Finally, the results are ranked based on the options selected by the user to provide the most relevant results to the user. Combination of these features forms a platform providing the first such complex and comprehensive service of this type.

### Challenges

The main challenges we have to face is recognition of user’s need based on their free text input. Further, we need to extract and bring together data from multiple sources using web crawling and APIs. Therefore, the task of extraction of the data for instance, finding the US doctors, as well as, connecting the data, for instance finding the specialization of a given doctor and which hospital he or she works for, are the most crucial and demanding tasks. Finally, the evaluation of the system requires careful investigation. The challenges technical challenges as well as the methods that we would like to apply to tackle them are presented in the section 4, together with the pipelines of the system development

## **3.3 IN4325 in our system**

In the back end of CareFinder, several information retrieval techniques are used to construct a database necessary to extract the demanded output. First, to find which doctor works at which hospital, web crawling is used to scrape doctor names from hospital websites. In order to find all the hospital URL’s, crowdsourcing is used.

To construct a search model for specialisation (mapping a search query to a specialisation), information from Wikipedia is extracted and filtered by tokenization, stemming and keyword extraction. To determine the expertise rank of a doctor, abstracts and keywords from papers and trials are extracted, tokenized and stemmed to build an index for search query-specialization mapping.

At last, the more generic thinking process of designing an information retrieval system based on user needs and available information is the core challenge discussed in the course and is executed in this project.

# 

# 

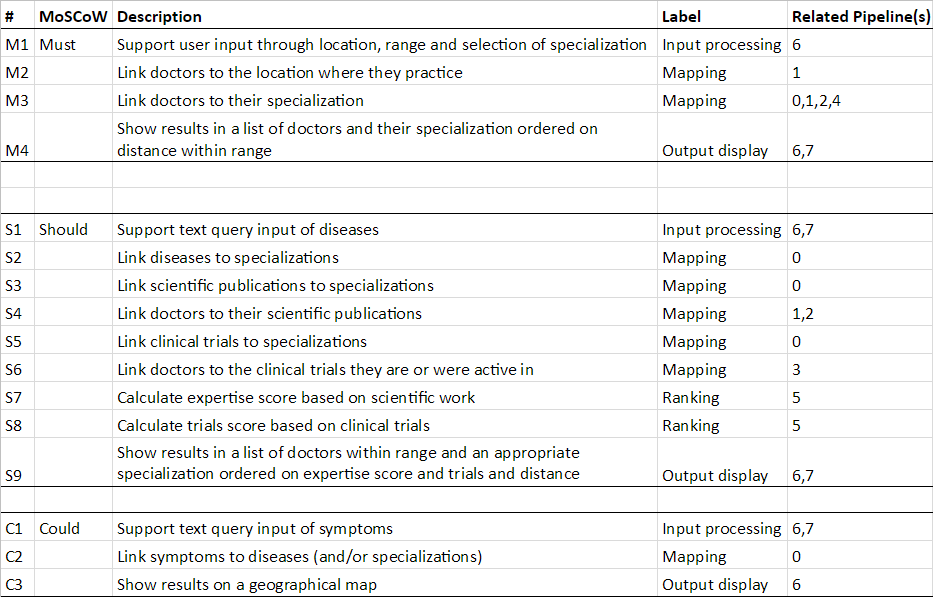
# **4. System design**

The aim of this section is to give a high level view on the planned development of the system. Even though the design might change significantly during the project, it presents valuable insights to understand the problem and our approach to solving it. However, before we dive into the real design of the system, we present the reader with a specification of the requirements of the project. Based upon these requirements, we have created several pipelines. The subsequent sections present these pipelines of development for the Indexing Process and the Query and Result Presentation[[6]](#footnote-5). Finally, we discuss the design structure of the database which will be filled in by the retrieved information and the indexes that will be used within CareFinder.

## **4.1 Requirements Specification**

Specifying the requirements for the final product at the very beginning of the software project will ensure that all developers and stakeholders share the same expectations. This will help developers to guide their design decisions and allow for an objective evaluation of the end results.

To that end, we are using the widely adopted MoSCoW requirements analysis, which separates requirements into 4 categories: Must, Should, Could and Won’t. One of the major points of criticism for this method is its lack of timing constraints (Wiegers, 2013). However, we will introduce an execution plan in the following section that should complement the shortcoming of the requirements analysis as presented here. Our analysis is based upon the problem description (See Appendix A) discussed in the first chapter as well as discussions held in the weekly coach meetings. However, since requirements will change over time in software projects (Wiegers, 2013), we do expect that our view on the requirements in the latest week of the projects may have become different. In the following section, we will elobare upon the actual design of pipelines in which these requirements are fulfilled. For your convenience we have already added a relation in the last column of the table below.



It is clear to see that the must have’s have been assigned to the requirements that - when implemented - together realize a minimum viable product, which basically provides a list of doctors that have a certain specialization and practice within a certain range of the user. However, the original assignment (see Appendix A), would require the should have’s to be fulfilled as well. These would adhere to a system that actually provides medical care based upon a subjective ranking. The could have’s correspond to requirements that are just nice to have, but will not have a significant influence on the evaluation of the system. We can also conclude that from their description, as they are more about making the application more convenient to use for the reader than providing better recommendations.

With regard to the won’t haves, we preferred to not to list them as specific requirements, but rather elaborate upon them (as they also may not be related to one of the pipelines in our design for instance). Hereby, we try to give the reader a feeling of what is outside the scope of this project. For now, these features are seen as potential future work that definitely will not be part of this project. In this project we will not:

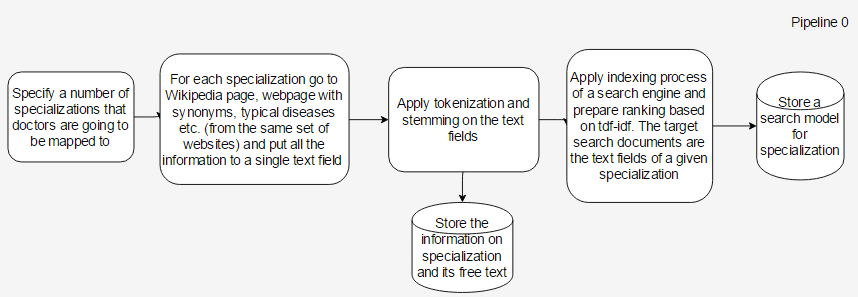
* Link directly to research papers or clinical trials
* Maintain a distinction between private, nonprofit or governmental hospitals
* Restrict geographical area by traveling time, the system is only able to give GPS coordinate distance
* Allow users to give feedback on the results they get, to further improve on the system

It is important to state that we do not say that an implementation of the above would not be valuable. However, we have chosen them to be outside the scope of this project as they are really an extension of the original assignment, and just like the could haves would not have much impact on the performance of the system with regard to providing the right medical care according to a location and a set of references (rather than e.g. budgets and travelling time).

## **4.2 Indexing Process**

The section presents the pipelines to be applied during the indexing process, to gather the necessary information for further querying. The Pipelines 0, 1, 2 and 3 show how we gather our data from different sources, then Pipelines 4 and 5 explain how we will find parameters used further in the querying process and finally the Pipeline 6 shows how we are going to develop our User Interface.

### Pipeline 0: Specialization Collection



**Dependent on pipeline(s):** None

**Approximate time:** 2 days

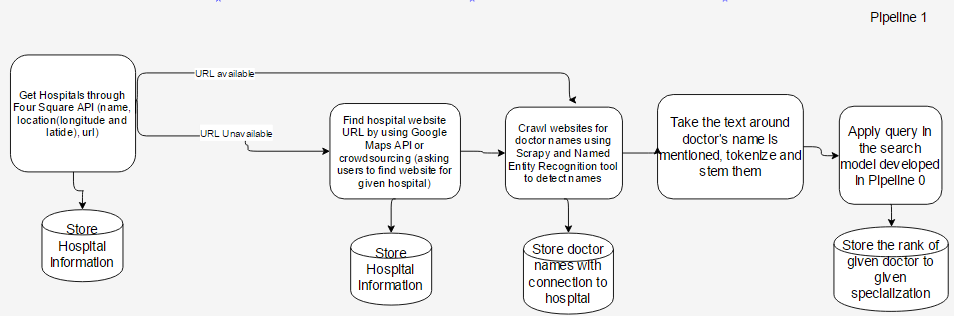
**Goal:** The goal of this pipeline is to fill in the data regarding different types of doctors’ s

pecializations. Moreover, we need to set up a document base for each specialization in order to further be able to do search by specialization. Finally, we have to conduct an indexing process to be able to query for doctor’s specialization based on free text.

**Steps:** The first step is identification of a number of possible doctor’s specialization, to do that, one has to search through the internet for a set of doctor’s specializations. Then, one needs to define on which websites we will look for the textual description of the specialization, synonyms and possible diseases given type of doctor handles. Then, from each of these websites we will copy the free text about the specialization e.g. description, typical diseases, synonyms etc. and concatenate it into one text field. The text field will be tokenized and stemmed, and after that the indexing process for a search system will be created. We suppose that the search engine should use the tf-idf ranking, because it handles well the words that appear in all documents as well as those that appear only in a small set of them. The engine will be widely used in the indexing and query process, to extract the rank for each specialization based on free text, that might be abstract of a research paper or even human input of a free text describing disease. Thanks to that approach we will be able to extract probabilities of specialization for any text input further in Pipelines 1,2,3 and 7. We will also omit the issue of medical sub-specializations, where one specialization belongs to another. In such case the search engine would rank them both highest since their textual description would be very similar.

**Challenges:** set up a search engine that can be used to rank how probable is that given text corresponds to given specialization. The engine should handle the free text input of a user, as well as more formal description like paper’s abstract.

### Pipeline 1: Hospital and Doctor collection



**Dependent on pipeline(s):** None

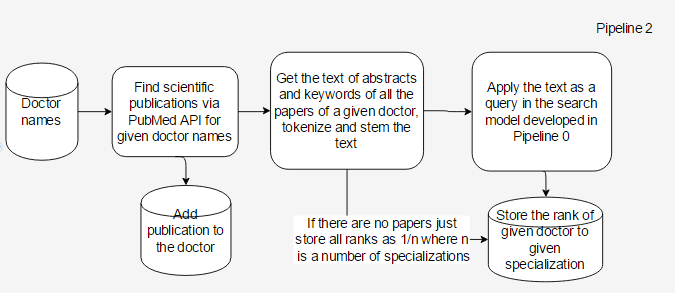
**Approximate time:** 1-2 weeks

**Goal:** The goal of this pipeline is to get the data of all the hospitals in the United States, together with all the doctors that work for these hospitals. Finally for each doctor we will rank how probable it is that he comes from given specialization

**Steps:** The first step is to get the data of the hospitals, together with the location and URL, which can be done via FourSquare API[[7]](#footnote-6). Then, for those hospitals that URL is unavailable, we plan to get the URLs by using crowdsourcing. We will make a csv file with the names of hospitals that we are missing the website for and their location and for each we will ask 3 people to provide the website for us on [www.crowdflower.com](http://www.crowdflower.com). Then a script will validate if at least two of the links match, else we will repeat the process. The information will be stored in the “Hospital” table of the database. The last step would be to program a crawler which extracts crawls through the pages of the hospital’s websites and extracts the names of doctors. For these we plan to use Scrapy[[8]](#footnote-7) for crawling on each page and Named Entity Recognition software, for instance the one Stanford Named Entity Recognizer[[9]](#footnote-8) for instance the one created by, for the detection of names in the website’s text. Whenever a person’s name is recognised we will add the person to the possible doctor set and if in the same sentence, up to 3 words before the name we find phrases like “doctor” or “dr.” these will confirm that the person is a doctor. Then we will take the text around the doctor’s name, e.g. 3 sentences after the name, tokenize and stem the text and use it as a query in the search engine set up in the pipeline 0. Then we will store the rank for each doctor and each specialization, based on the website’s text

**Challenges:** The main challenge is crawling the websites, since many of them have different structure and multiple of them contain automatically generated content, instead of static websites. Moreover, the task of detecting whether a possible doctor is actual doctor is non-trivial as well, since on many websites the “doctor” phrase might not be present. In such cases we would need to investigate possible alternatives detection.

### Pipeline 2: Specialization extraction based on papers



**Dependent on:** pipeline 1

**Approximate time:** 1 week

**Goal:** The goal of this pipeline is to extract publications for the set of doctors, then to rank for each doctor the specialization that he has based on the abstracts and keywords of his papers.

**Steps:** The first step is the extraction of the papers, their abstracts and keywords of the papers of given doctors. Knowing the full name of a doctor, we can query by a surname and then validate if the first name or the first letter matches the person. Then, once we gather the set of publications and their abstract for all the doctors, we would like to gather all the paper’s abstracts and keywords of given doctors’ papers into a single text field, apply tokenization and stemming on it and finally by the use of search engine to conclude the expertise of a doctor belongs with regard to a given specialization.

**Challenges:** The main challenge of this step is validating whether the ranking process works well for different specializations, since the papers might be really specific and use scientific language, as well as distinguishing John Smith (who published X) from John Smith (who published Y). We can overcome the challenge by extracting from the research paper, the city name of it's publishing or the names of the cities from the first page and assigning the paper to the doctor that works the closest to the most frequently appearing location.

### Pipeline 3: Specialization extraction based on trials

### 

**Dependent on:** Pipeline 0 and 1

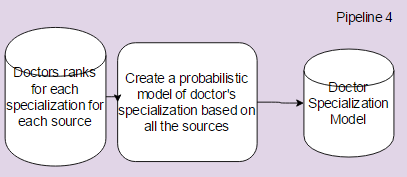
**Approximate time:** 4 days

**Goal:** The goal of this pipeline is extracting the medical trials information of the doctors as well as their ranking in terms of specialization based on the text of the clinical trials

**Steps:** The first step is extracting the clinical trials using clinicaltrials.gov API[[10]](#footnote-9) for all doctors based on their surname and by validating if the name or the first letter of it if only this information is available matches. Then, on the text of all the trials of a given doctor we will apply tokenization and stemming and use the search engine from the Pipeline 0 to rank which specialization is the most probable for the person.

**Challenges:** As in the Pipeline 2 the main challenge is finding out whether the search engine works well in recognising specialization based on the free text of a clinical trial. Moreover, as before, the problem would be if two doctors have exactly the same name and surname, how to recognise which one of them conducted the trial. As before the challenge may be solved by extracting the names of the cities from the trial information and assigning the trial to the doctor that works the closest to the most frequently appearing location.

### Pipeline 4:



**Dependent on:** Pipeline 0, 1, 2 and 3

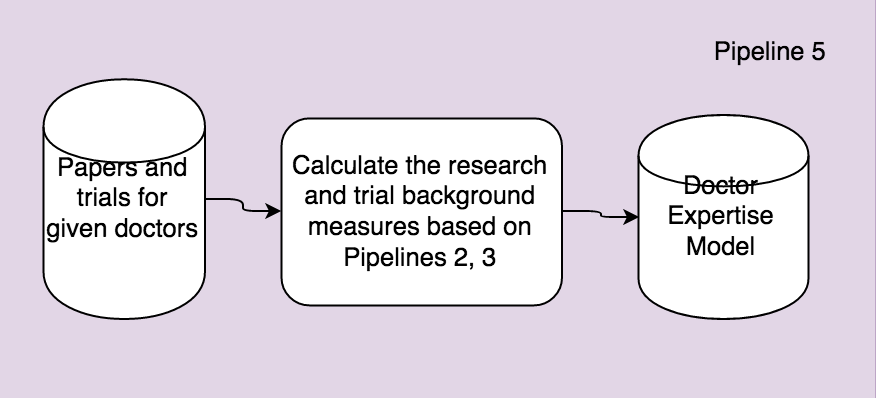
**Approximate time:** 2 days

**Goal:** The goal of this pipeline is creating doctor’s specialization model, ranking of doctor to the specialization as acquired from different sources

**Steps:** In this section we gather results from Pipelines 1, 2 and 3 in terms of ranking to a specialization of each doctor from different, and most probably simply multiply them by each other for each specialization. Then we will normalize the results and it will give us the probability that a given doctor belongs to a given specialization

**Challenges:** Validation whether the model gives reasonable results

### Pipeline 5



**Dependent on:** Pipeline 0, 1, 2 and 3

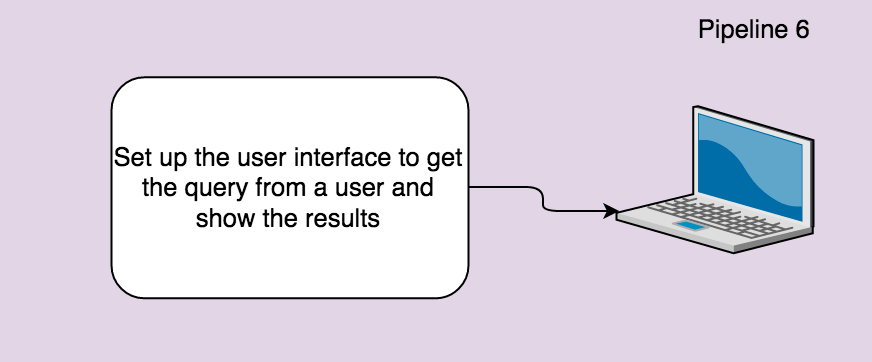
**Approximate time:** 2 days

**Goal:** The goal of this pipeline is calculating for each doctor how high his score is in terms of the amount of scientific papers and clinical trials he took part in.

**Steps:** Firstly we get for each doctor all the research papers, we count those and normalize. Finally, each doctor has a paper and trial score from 0 to 1

**Challenges:** Normalization in this step is trivial, yet, making a ranking function that takes them into consideration and gives good results will be a challenge.

### Pipeline 6



**Dependent on:** None

**Approximate time:** 2 days

**Goal:** The goal of this pipeline is creating a User Interface, which gathers the necessary information for a successful query.

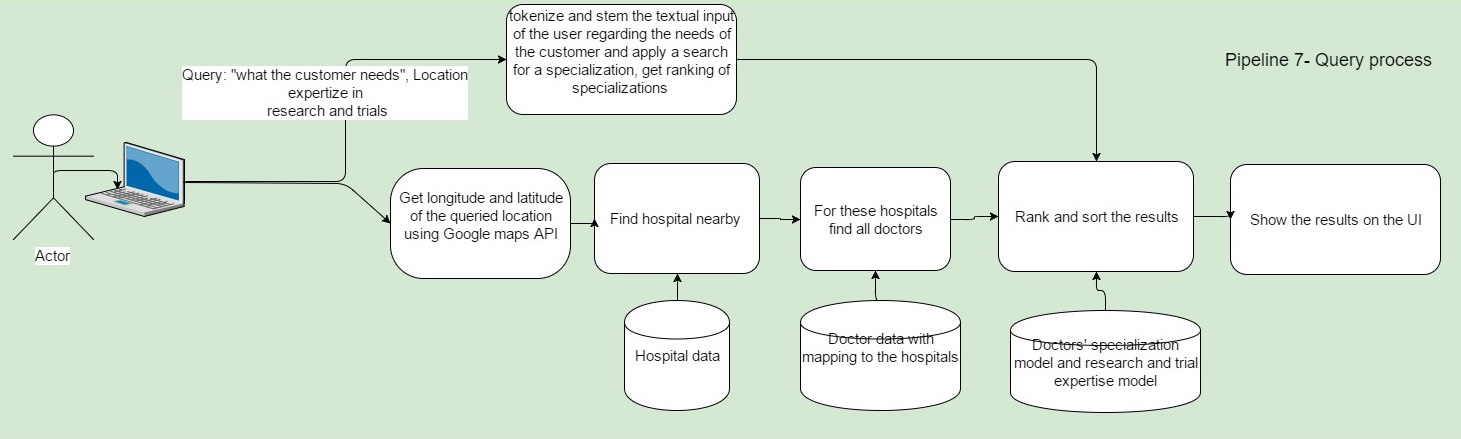
**Steps:** The only step of this section is designing and implementing a user interface with the fields that gather user’s input and the panel showing the results of the search. Optional step is using Google Maps Api to show the results on the map. The interface should to choose what is the minimum research and trial background.

**Challenges:** None

## 

## 

## **4.3 Query and Result Presentation Process**

**Dependent on:** Pipeline 0, 1, 2, 3, 4, 5 and 6

**Approximate time:** 2 weeks

**Goal:** The pipeline should allow users to query the system for the doctors, based on his/her needs.

**Steps:** Firstly we get the query, then the flow splits into two directions, first one based ranks each specialization based on the input of the user, the other one retrieves hospitals nearby and finds doctors that work in those. Then, those flows connect, ranking and sorting process based on the specialization rank, doctors and their expertise is applied and the results are shown to the user

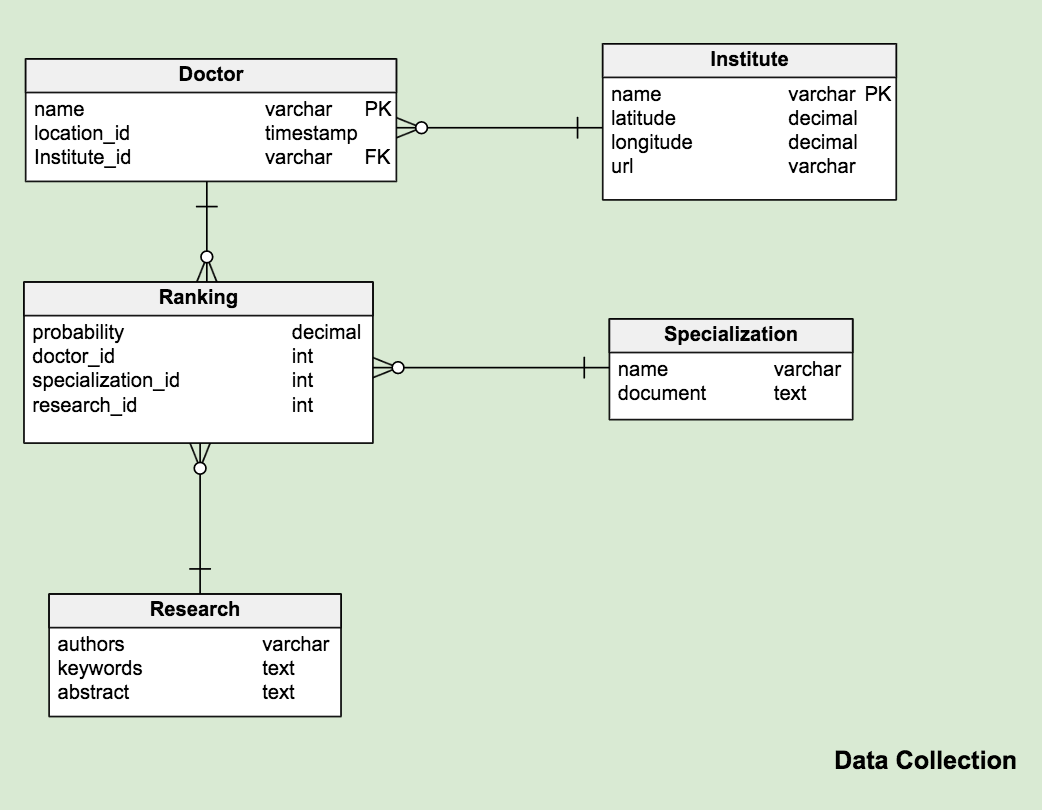
**Challenges:** Ranking of the results in a correct way, to get a high precision and recall.

## 

## 

## **4.4 Data Model**

The section focuses on the Database structure which supports our project. The graph below, presents all the necessary data tables and connections between them.



### Doctor

The records in this table correspond to the doctors, who are our main subject. A reference to the Institute table references where a person is working. Furthermore, each doctor has a rank for each specialization and each source of this information(website, papers, trials, final).

### Institute

The institute is a location where doctors are executing their profession. This can be of several types such as Hospital, University, Lab. Since a categorization of those is not necessary at this point, we agreed upon describing an institute by a name which generally should be sufficient to describe its type – e.g. Massachusetts Institute of Technology, Manhattan Psychiatric Center, etc.

### Ranking

The ranking table creates a mapping between the doctor, a specialization, and the underlying research. Essentially, the table stores ranking of each doctor mapped to each specialization based on 3 main sources of this informations, website, research papers and clinical trials.

### Specialization

A simple collection of medical fields. In addition, we store the content of all related documents for a specialization, concatenated in a text field. The sources based on which we create the documents should include the general information, synonyms, related diseases.

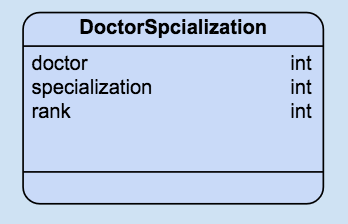
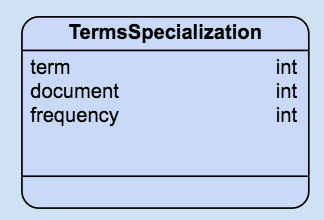
### Research

The research table holds the content of the collected research. This can be of various categories. As we are interested in the content solely, we do not have to apply categorization.

## **4.5 Index tables**

The section describes the tables necessary to store the data used directly by the search engine of Pipeline 0, namely Terms Specialization table, as the probabilistic model of doctors’ specializations used in the Querying process.

### Terms and Doctor Specialization



In order to associate the terms found in the documents for specializations, we create a view (see image on the left) that holds all the terms of the entire set of documents and see how often this term appears in a document. Provided that the document is already linked to a specialization, a search engine will now be able to create a query with trivial terms and retrieve the corresponding specializations with a rank.

The second index holds doctors and associates a specialization with a rank – which is basically another representation of the “Rank” table (see Data Model). However, while looking a doctor (or a specialization), this immediately returns the specialization (or doctor).

# **5. Planning and Evaluation**

Since we are making a software product, we also want to create an execution plan and means to evaluate the performance of our system. In this section we elaborate upon both separately in the following sections.

## **5.1 Execution plan**

We started our execution planning by calculating the minimal available amount of hours of each team member, to get a feeling of how much time we have (at least). Afterwards, we created an overview of the execution planning for the w

### Minimum time per member

The IN4325 course is specified to have 5 EC in the European Credit Transfer System. Each of those are good for 28 hours of work in the Netherlands\footnote{See \url{https://en.wikipedia.org/wiki/European\_Credit\_Transfer\_and\_Accumulation\_System}}. The course involves 2 weekly lectures of 2 hours each, 1 weekly individual assignment and 1 peer review of 30 minutes each and 1 coach meeting of 30 minutes, as well as a final individual assignment which was by the lecturer estimated to be 3 full-time days of work.

5 EC = 5 x 28 hours in total = 140 hours

1x Individual final assignment (estimated to be 3 full days of 8 hours) = 24 hours

10x Lectures, individual assignments and peer reviews, coach meetings = 55 hours

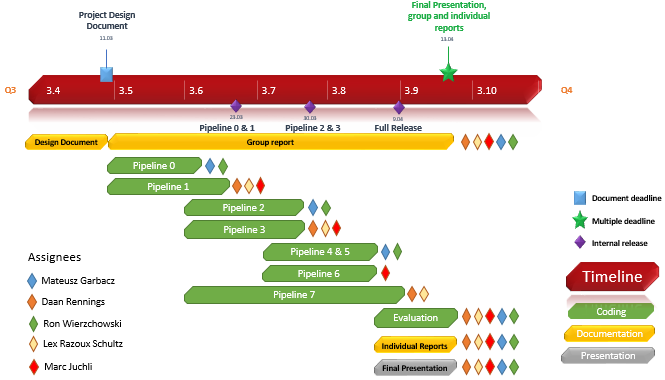
Resulting in 61 hours per team member left to work on the project in total (10 weeks), leaving about 30 hours to be spent over the upcoming 5 weeks. We therefore concluded that the minimum input from each team member would have to be 6 hours per week.

### 

### 

### Planning

The graph below presents the the plan of Carefinder development.



As shown in the image, we will have 3 major internal releases before the final deadline. Each release will focus on the most important parts of the system that can be developed due to requirements of the system and are also necessary for other parts to start. We plan to assign to each demanding pipeline at least two people and in the end work together on the evaluation, presentation and documentation of the project.

## **5.2 Evaluation**

Next to the importance of having the requirements of a complete system and a design and planning to realize it, it is just as important to evaluate the performance of the system. Since the focus of this project is on information retrieval, we have decided to leave out aspects like process evaluation and evaluation of the implementation (e.g. by using Sonarqube or do a usability test with real life users). In this section we therefore present the reader with product evaluation, in which we focus on evaluating the results of our care recommender system by precision metrics, the adherence of the system towards the requirements as specified in section 4.1 and how well our system performs compared to the related work called DoctorFinder.

### 

### 

### Product Evaluation

#### Evaluation metrics

There are various metrics that could be used for evaluating the end product. However, we are limiting ourselves to investigating the following three popular metrics. By testing all 3 of them we are hoping to learn which method or combination thereof is best suited for this task, as they have various pros and cons, just like various users may have various preferences (e.g. precision over recall or the other way around). We are foreseeing this variation in preferences as it is about medical care. Anxious people might want to find the one very best doctor in the whole of the US, whereas others might want to see as much options as they can, since factors such as pricing (think of the governmental and private hospital distinction not made by our application) which our system does not take into account.

***Precision@K*** is a simple evaluation metric that calculates the precision up to a given rank K. It is easy to implement though not very flexible.  
  
***Mean Average Precision*** calculates the P@K for each rank position K of relevant documents. It gives the average precision across multiple queries and rankings.  
The underlying assumption is that users want to retrieve many relevant documents with each query. It is frequently used for example in scientific paper recommendation systems (Tang, 2012). However, this method requires more relevance judgements per item to achieve meaningful results.  
  
***Normalized Discounted Cumulative Gains*** aims to evaluate the usefulness of results by assuming that the most relevant documents should be ranked highest and medium relevant documents should be ranked above non relevant documents in order to maximimize usefullness. This method requires precise relevance judgements that have to be much more fine grained than binary gradings (relevant/ non-relevant worked well for the first two metrics). This metric is best when trying to give few or even only one (best) recommendaftion.

So far it seems that the only feasible way for us to get a test collection will be to give relevant judgements ourselves and/or use crowdsource this task. However, we also may use the results of a similar application (as discussed in the final paragraph of this section), however we would still have to manually rank the results ourselves. For initial measurements on the other hand, we think that scanning through the results would already hint at the performance of the system.

#### 

#### 

#### Requirements Specification

Another part of evaluating the resulting software product will be to check which parts of the requirements specified in section 4.1 have been fulfilled. According to the MoSCoW analysis all Must requirements that have not been fulfilled will definitely have a negative impact, while all Should and Could specifications can be evaluated as small positive contributions. It is important to keep in mind that requirements can change, (although that is less likely at the scale of this project), when reflecting on the initial requirements.

#### Comparative Study

In the initial exploratory phase of this project, we have come across an application called DoctorFinder[[11]](#footnote-10). On their webpage, this application hosted by the American Medical Association is defined as an online physician Locator that *“... helps you find a perfect match for your medical needs DoctorFinder provides you with basic professional information on virtually every licensed physician in the United States. This includes more than 814,000 doctors.”*. The application is similar to ours, as it also asks the user to input a location and specialty (or name of a physician they are searching). However, our application goes beyond the capabilities of DoctorFinder in the sense that it also ranks the doctors matching the input and may also allow the user to insert a search query instead of selecting a specialty. We will compare the results of our project with the DoctorFinder application, to be able to have a guess at the recall of our system, which can already be done after having realised the first fully operational system (consisting of fully implemented must haves).

# 

# 

# 

# **6. Bibliography**

Wiegers, Karl, and Joy Beatty. Software requirements. Pearson Education, 2013.

Tang, Jie, et al. "Cross-domain collaboration recommendation." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

# 

# 

# **Appendix A. Original Project Description**

## **Project 1: CareFinder**

Owner: <https://mytomorrows.com> & Web Information Systems (Alessandro Bozzon)

To date, no Web search engine allows to search for hospitals and clinics by the type of specialist care they offer. This is information typically available on an hospital’s Website, but not structured in such a way to allow exploration. Also, there is little information about the quality of the doctors, for which some information could be available on-line. For instance, by looking in online repositories for articles published by doctors of the hospital (e.g. on <https://www.ncbi.nlm.nih.gov/pubmed>) , or for ongoing clinical trials (e.g. https://clinicaltrials.gov)

**Assignment**: design and implement a sensemaking system that, given a country or region of references, scouts from social data (e.g. FourSquare) hospitals, identify their public information (e.g Website), extract the list of disciplines and related doctors, and provide a measure of quality of care based on the scientific productivity such doctors, and on the presence of active clinical trials.

1. Stats.oecd - <http://stats.oecd.org/Index.aspx?DataSetCode=SHA> (retrieved March 2017) [↑](#footnote-ref-0)
2. S[tatista](http://www.statista.com) - <https://www.statista.com/topics/1244/physicians/>, <https://www.statista.com/topics/1074/hospitals/> (retrieved March 2017) [↑](#footnote-ref-1)
3. Clinicaltrials - <https://clinicaltrials.gov/> (retrieved March 2017) [↑](#footnote-ref-2)
4. See https://www.stephencovey.com/7habits/7habits-habit2.php [↑](#footnote-ref-3)
5. AMA Doctor Finder, website, where users can find doctors based on the location and specialization - see <https://apps.ama-assn.org/doctorfinder/recaptcha.jsp> [↑](#footnote-ref-4)
6. See also lecture slides, IN4325, “Information Retrieval”, week 1 [↑](#footnote-ref-5)
7. FourSquare API is a software allowing to extract data from the FourSquare website- see https://developer.foursquare.com/ [↑](#footnote-ref-6)
8. Scrapy is a software allowing to apply web crawling and web scraping on given websites, see https://scrapy.org/ [↑](#footnote-ref-7)
9. Stanford Named Entity Recognizer is a software allowing to recognize phrases in a free text, see http://nlp.stanford.edu/software/CRF-NER.shtml [↑](#footnote-ref-8)
10. Software allowing to extract data from the website clinicaltrials.gov regarding the clinical trials and doctors that took part in them [↑](#footnote-ref-9)
11. See https://apps.ama-assn.org/doctorfinder/ [↑](#footnote-ref-10)