TDT4215 Web-Intelligence Project

Magnus L Kirø
Vuk Ilic
Filip Jankovic

April 25, 2013

Wordcount: 3174

Abstract

A web intelligence project to learn about searching and data gathering.

Contents

Introduction

The goal of the project is to connect information from *Legemiddelhåndboka* to 8 given patient cases. To acchieve this we have to use some smart libraries to parse data, and some kind of indexing and search algorithms.

The intelligent part of the project are the dataparsing. Seeing patterns and usefull metadata in the datasets. And the indexing and searech with the different types of metadata that are available.

System architecture

Top level description

The system is divided into four packages. Project, Parsing, Models and Search. The project package is the main package. Here the main files lie.

The main class is "App". The app class creates a new CaseController. The casecontroller then imports all the patient cases.

Then we create a new Parser. The parser reads all the datafiles. All the html chapters of *Legemiddelhåndboka*, the atc's and the idc-10's. The parser returns a new SearchEngine object with all available data.

The searchEngine object is registered in the CaseController.

When a new SearchEngine is created the indexing and searching is done. This is where we link all data towards sentences in a patient case or to just to the patient case in itself.

Now that the initialization is comoplete we can ask for a case to be displayed. Using the "case x" command, we ask that patient case x is displayed.

The displaying is controlled from the CaseController. for simplicity each case has it's own display method that pronts the given content.

Architectural traits

The system has some resemblance to Model and View Controller (MVC). While not explicitly called controllers we have classes executing operations to create models. And then display the information in them.

The whole model packages describes data objects that are used in the system. While everything else are a controller.

The views come into account from the CaseController where we actually display the data stored in the models. Although the viewing part is minimal it is still present.

Components

The system has three main modules. CaseController, Parsing and Searching.

App

This is the main class. It initiates the other classes. Then loops infinitely for terminal input.

It derives commands from the terminal input. Such as "case 1", which displays the data for case 1. its codes, linked chapters and such.

CaseController

The case controller is the last instance that we need. When all data parsing, document indexing and searching is complete we only need the cases and the data it contains.

Search module

The SearchEngine has four sub classes. This is to simplify the search process. At present the SearchEngine is created with lists off all the data objects. Then it calls the different search classes to index and link all the given inforamtion.

ATCSearch

The ATCsearch takes the cases and all the atc codes. Then we use all the atc's as documents for the lucene index, and add them there. Proceeding we take all the sentences of each case and uses taht as a search query to get relevant atc objects for each sentence. Then we store the top four atc objects in the sentence.

ICDSearch

We do pretty much the same of ICDs as we do for the ATCs. we index all the ICD codes on label and uses the sentences of the patientcase to search the index for the most relevant codes. Then store them in the sentence.

ChapterSearch

The ChapterSearch is used for the three kinds of chapters, therapy, general and the drug chapters. As the in the other search classes we do this for all

the cases. We use the sentences as queries to search the index. The index is consisting of all the paragraphs in chapter. If the paragraph is matched we register the chapter in the sentence. This way we keep track of which chapters that contain the most relevant information for the given sentence.

FullCaseSearch

The FullCaseSearch takes all sentences in a case and builds one query for the whole patient case. Then we use this to search all chapters after the most relevant ones.

Parsing module

The parsin module also consists of sub classes. Here we have the same division as before. The parsing is sepparated in to ATC's, ICD's and chapters.

HTML parsing

The chapters are parsen in the HTMLParser. The HTML parser is based on Jsoup. A library that reads HTML files and get certain elements from the content. We figured out that all the chapters have a certain structure.

The toplevel chapters contain no usefull information, while the second level chapter files contain all the information we need. So we skip chapters in the form 'L17.htm' and only considers 'L17.X.htm'.

In the sub chapters we have sub sub chapters. All the sub sub chapters are described in the menu of a sub chapter. So we extract the div ids from the menu to be able to only extract the specific parts of the html document.

When we know of all the sub sub chapters in a sub chapter. We create chapters of them, and link the parentage.

To get tha paragraphs of a sub sub chapter we look for the h5 element as the title, and the p element as the paragraphs for this sub sub chapter.

ATC and ICD parsing

The atc and ICD parsing are actually in different classes, but they are so similar that there isn't a point to have multiple descriptions of them.

So we have the .owl file format to work with. To be able to read this type of file we use the jena library. This makes the reading of owl files doable and functional.

Whitout the library the best way to read the file would be some kind of regex based line by line parsing. Which would have been tiering and unnecessary.

The library loads the file the .owl file into memory, and creates ontology classes. These classes aren't much useful for us, so we have to extract information from these ontology classes.

We do this by using properties. The Property object of the jena library gives us an anchor for the wanted information in the given ontology class. We specify the properties we want. And get store all the wanted properties of an ontology class in our own corresponding class.

When all the owl classes have been created we link the parentage of the objects. This gives us the ability to get classes and subclasses of any object. Expanding our future functional posibilities. Like viewing the next chapter from the current one. Or traversing the tree of objects.

Case parser

The case parser in simple. It runs through the file containing all the patient cases. Reading every line. If the line contains 'Case x' then we create a new case object. else we read the line, split the lines into sentences, and adds them to the case object.

Utils

The utils class have common static methods that are used in the other parsers. This is to create modularity and ease of use.

It also makes the code a bit cleaner.

The models

ATC

Are the atc codes decoded from the atc.owl file. These are medical drugs with sub and super classes.

A kind of hierarchy of drugs.

ICD

The ICD-10 codes describe different decieases. We store it's label, it's id code, its parentage, and the synonyms stored in the ontology class.

The synonyms and the parentage should have been used in the searching to get better results.

Chapter

These correspond to chapters in the *Legemiddelhåndboka*. We parse them, store the paragraphs and the filename as id. We also store the title and parentage of a given chapter.

With the parentage we can potentially travese a tree to a given sub chapter. Or otherwise use the tree structure to our advantage. The only problem with that is that we disregard the top level cahpters.

Paragraph

Paragraphs correspond to p elements in the html files. We store them so we can easily get sub parts of a chapter.

Case

The patient case. Consisting of the case id and the sentences.

It alsto has a list of chapters that is directly linked to it. These are the chapters that combly the best with the whole of this chapter.

Sentence

The sentences are basically an object containing a string and an id. The sentences also contain the linking towards the other datatypes. The chapters and the medical codes.

The linking of data objects are done with hash maps. The top four resulting documents are stored in that order in the hash map. THe rank of the object is its key.

Limitations

The task in itself gives few limitations. And that's a problem. The lack of limitations on the task gives us the possibility of doing pretty much anything we want and still have the task comply to the task description.

The loose wording of the task description and the usage of expresseions along the lines of 'might' and 'can be an idea', increases the compliance of product and description. No boundaries or definitions means that anything will be accepted. Which limits us greatly to what we think that task description actualy wants us to do.

Setting aside the task description the biggest limitation was the time frame and team effort. These limitations were present mostly becasue of

other subjects interfering with available time. The group met once, when we should have met a lot more then that.

Techical limitations occured with the owl file. The jena library couldn't handle more then one subclass. Or duplicate properties in the same class. So a calss that is a subclass of two classes are only parsed as a sub calss of the last mentioned superclass.

The lucene library works well. Although we should have used some more time to tune queries and probably spent some more time improving the indexing part.

Theory

The project incorporates some theory of web intelligence and search techniques for text documents. Whether the theories of the subject are incorporated in the libraries used or not is beyond the scope of the project. Althought they proably are.

Other theories of techniques that could have been utilised could have been clustering, stemming, more complex text minig and linguistics.

As a required point of the report this point should have been better defined. Which theories are we talking about? Quantum mechanics?

Results

We have results. But I don't think that the results are 100% accurate. The best we can confirm that our results indicate is that we have managed to link the chapters of *Legemiddelhåndboka* and the patient cases. We have also shown that the apache lucene library can be used to index text documents and searching in them afterwords.

The result output displays the case name and id. The ranked chapters for the full document. All sentences with id and content. All ranked objects for each sentence.

The objects are comma sepparated.

Following we have example output. For the 7 other cases, run the code.

```
----------------------- Ready -----------------------
case 1
Finding current case: 1
case: 1
fullCase: G12, G14, G7, T8.1,
0: Eva Andersen er en skoleelev som har hatt insulinkrevende diabetes
```

mellitus i 3 år
ICDs: H280, E10-E14, P702, Z833,
ATCs: V04CA, A10, G03BA, G03BB,
G: G24, G10, G22, G16,
L: L17.2, L1.1,
T: T10.11, T19.7, T8.1, T5.10,
1: Hun har en bror som også har diabetes og som har brukt insulin i flere år
ICDs: E10-E14, E232, Y631, Y882,
ATCs: V04CA, A10, A10X, DB01309,
G: G10, G16, G3, G17,
L: L17.2, L1.1, L7.1,
T: T8.1, T19.7, T1.9, T8.12,
2: Hun bruker insulinpenn med hurtigvirkende insulin som hun injiserer under huden før hvert måltid og dessuten en injeksjon langtidsvirkende insulin om kvelden
ICDs: Y613, Y613, Y613, D485,
ATCs: DB01309, A10AB01, A10AB02, DB00046,
G: G14, G12, G2, G8,
L: L7.1, L1.1, L17.2,
T: T10.10, T19.3, T19.1, T1.9,
3: Hun har vært lite motivert for selv å håndtere sin sukkersyke
ICDs: C511, P051, N270, O261,
ATCs:
G: G9, G6, G19, G8,
L: L7.1, L17.2, L1.1,
T: T5.5, T5.9, T1.11, T8.1,
4: Hun uteblir fra kontroller, har delvis ikke tatt insulin og periodevis ignorert kostråd
ICDs: Z971, Z972, Q263, Z35,
ATCs: DB01309, A10AB01, A10AB02, DB00046,
G: G7, G12, G18, G6,
L: L17.2,
T: T1.13, T8.1, T8.12, T23.6,
5: Hun synes det er leit å ha fått sukkersyke
ICDs: R192, H541, Z010, H542,
ATCs:
G: G10, G14, G20, G21,
L: L7.1,
T: T23.2, T1.11, T19.3, T4.2,
6: Eva har siste døgn følt seg tungpusten, vært kvalm og kastet opp
ICDs: R11,

```
ATCs:
G: G12, G23, G2, G15,
L: L1.1, L17.2,
T: T19.3, T5.10, T19.2, T1.2,
7: Rødmusset
ICDs:
ATCs:
G:
L:
T:
8: Hurtig pust
ICDs: O366,
ATCs:
G: G2, G12, G5, G7,
L:
T: T19.5, T1.11, T1.9, T1.8,
9: Tørr i pannen
ICDs: E500, E502, E501, O323,
ATCs:
G: G10, G12,
L:
T: T16.1, T14.3, T19.1,
10: Acetonlukt
ICDs:
ATCs:
G: G12,
L:
T:
11: Normalt blodtrykk.
ICDs: Z013, Z340, Z34, Z348,
ATCs:
G: G16, G14, G6, G12,
L:
T: T8.1, T1.10, T8.12, T1.9,
12: Hun er blitt delvis uklar, og vurderer henvisning til sykehus
ICDs: Z380, Z383, Y872, Z381,
ATCs:
G: G12, G23, G20, G18,
L:
T: T19.6, T1.10, T5.9, T19.1,
-----------------------CMD END -----------------------
```

Evaluation and discussion

The result have not been thoroughly tested and verified. We have results. The program works. The accuracy of the program should be improved. Specially with the indexing of documents.

The project in itself is good. But the description and the actual goal of it should hade been accomplished better.

We could have automatically tested our result toward some correct standard. As we save all the data and link all the objects to cases this is a small task. Although the standard ain't very compatible to the format of our results. The time to alter the standard to fit our results were not taken.

Potential improvements

The emmediate imprivements that we could have done would be stemming of words used in the indeing phase. And further expanded our queries to multi field queries, to utilise the metadata of our objects. And to better use the titles and names of files.

Task Critics

The task description is poor. There are no defined subtasks that has to be completed. The overall objective is unclear. And the ranking standards are not set.

Also the document should be provided in a pdf format for ease of use. The task description should also be updated with corrections througout the course.

Conclusions

Overall the project is quite good considering the time spent working on it. Although the actual result is far from the best work any of us have done. There are a lot of improvements that could have been done to increase correctness and computaion time.