Automatic ICD-9-CM Classification for textual medical reports

Margherita Lazzarini, Massimiliano Fasi[1]

11 febbraio 2013

[1] {margherita.lazzarini, massimiliano.fasi}@studio.unibo.it

Abstract

An algorithm for automatic **ICD-9-CM** classification is presented, as well as a working though experimental implementation.
The interface returns to the user a set of possible classification for a given textual description of the diagnosis. The user has to select the most suitable classification. Multiple classifications for a same session are allowed.

Tools

We used the following software frameworks:

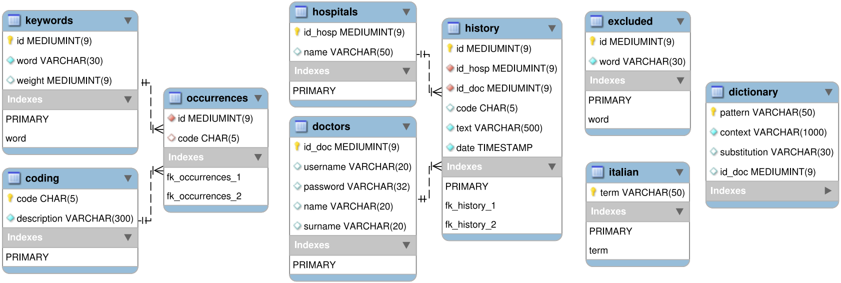
- Server side
    - *Java* servlet within *Tomcat* container on `golem.cs.unibo.it`
    - *jdbc* Java-based data access technology
    - MySQL database on `golem.cs.unibo.it`
    - *Python*

Tools

We used the following software frameworks:

- Server side
  - *Java* servlet within *Tomcat* container on `golem.cs.unibo.it`
  - *jdbc* Java-based data access technology
  - MySQL database on `golem.cs.unibo.it`
  - *Python*
- Client side
  - *HTML* and *CSS3*
  - *AJAX* technologies
  - *JQuery* framework

Database structure



keywords
id MEDIUMINT(9)
word VARCHAR(30)
weight MEDIUMINT(9)
Indexes
PRIMARY
word

coding
code CHAR(5)
description VARCHAR(300)
Indexes
PRIMARY

occurrences
id MEDIUMINT(9)
code CHAR(5)
Indexes
fk_occurrences_1
fk_occurrences_2

hospitals
id_hosp MEDIUMINT(9)
name VARCHAR(50)
Indexes
PRIMARY

doctors
id_doc MEDIUMINT(9)
username VARCHAR(20)
password VARCHAR(32)
name VARCHAR(20)
surname VARCHAR(20)
Indexes
PRIMARY

history
id MEDIUMINT(9)
id_hosp MEDIUMINT(9)
id_doc MEDIUMINT(9)
code CHAR(5)
text VARCHAR(500)
date TIMESTAMP
Indexes
PRIMARY
fk_history_1
fk_history_2

excluded
id MEDIUMINT(9)
word VARCHAR(30)
Indexes
PRIMARY
word

italian
term VARCHAR(50)
Indexes
PRIMARY
term

dictionary
pattern VARCHAR(50)
context VARCHAR(1000)
substitution VARCHAR(30)
id_doc MEDIUMINT(9)
Indexes

Figure: General ER Schema of the database

ICD-9-CM core tables

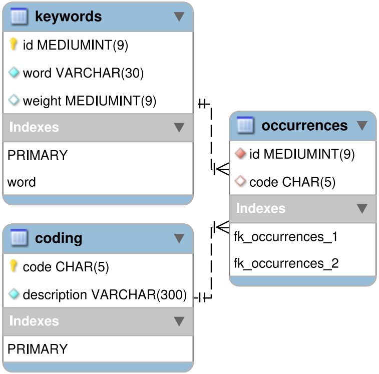


Figure: Detail of the ER Schema

- **coding** contains the ICD-9-CM diseases classification
- **keywords** contains every significant word of the classification with the corresponding weight
- **occurrences** links each keyword to the codes whose description contains it

ICD-9-CM history management tables

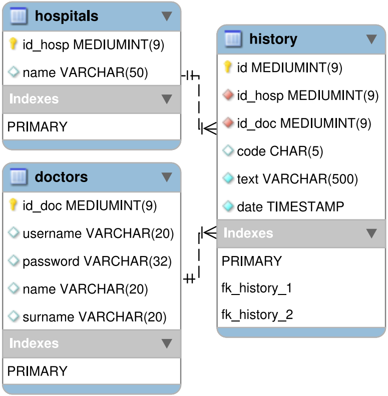
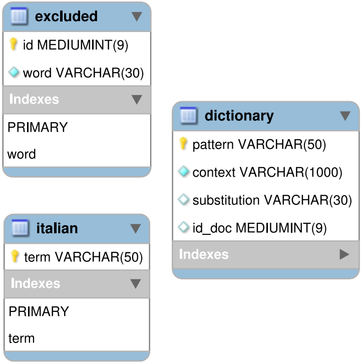


Figure: Detail of the ER Schema

- **doctors** contains the personal and login information of each authorized user
- **hospitals** contains the information about each hospital using the service
- **history** links the doctor with the textual diagnosis, the diagnosis time and the chosen ICD-9-CM class

Text parsing and mining tables



excluded
- id MEDIUMINT(9)
- word VARCHAR(30)

Indexes
- PRIMARY
- word

dictionary
- pattern VARCHAR(50)
- context VARCHAR(1000)
- substitution VARCHAR(30)
- id_doc MEDIUMINT(9)

Indexes

italian
- term VARCHAR(50)

Indexes
- PRIMARY
- term

Figure: Detail of the ER Schema

- **excluded** contains the list of *stopwords* to be replaced during the normalization step
- **italian** is a complete list of Italian terms produced expanding a context-free grammar
- **dictionary** contains shortenings widely used in medical context, along with the corresponding expansions

Database's population

The initial state
Ab ante, the database is populated by *Python* scripts that parse *comma separated* files (*.csv*) containing the classification, the Italian dictionary and a given set of past years diagnoses.

Database's population

The initial state

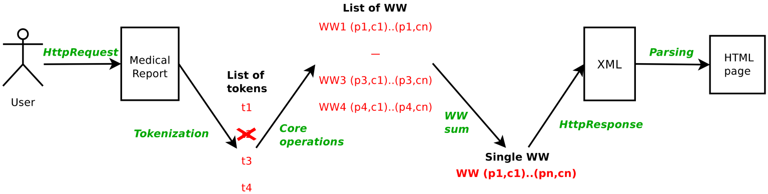
Ab ante, the database is populated by *Python* scripts that parse *comma separated* files (*.csv*) containing the classification, the Italian dictionary and a given set of past years diagnoses.

The state's update

During the execution, many of the tables are modified by the *Java* servlets:

- **dictionary** is updated both when new diagnoses are inserted and when new expansions of the shortenings are manually (by dedicated servlet) inserted
- **history** is updated just when a new association between diagnosis and classification is sent to the system

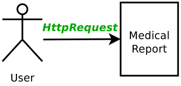
Algorithm's outline



The string inserted by the end user is split into **tokens**, and each one of them, if considered significant, is turned into an object of the class $WeightedWord$, that is closed under a structured **sum** operation.

By successive sums, a single object of that class is created, and the output is sent back to the client through an *XML* string, **parsed** by an appropriate *JavaScript* function.

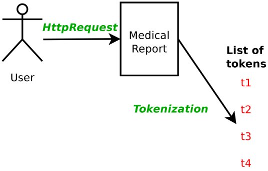
Tokenization and Normalization



The medical report, sent through a *GET* request to the servlet, is divided into single words, that, if not recognized as *stopwords*, are normalized as follows:

- everything is lower cased,
- accents, if any, are removed,
- tokens that matches one element of the PATTERN column of the DICTIONARY table, are replaced accordingly to the corresponding SUBSTITUTION.

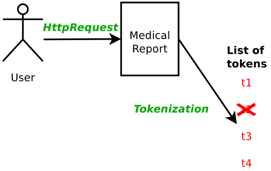
Tokenization and Normalization



The medical report, sent through a *GET* request to the servlet, is divided into single words, that, if not recognized as *stopwords*, are normalized as follows:

- everything is lower cased,
- accents, if any, are removed,
- tokens that matches one element of the PATTERN column of the DICTIONARY table, are replaced accordingly to the corresponding SUBSTITUTION.

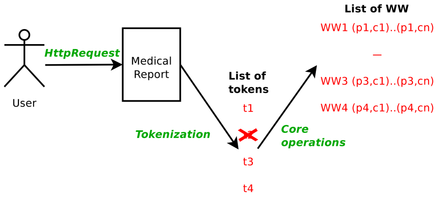
Tokenization and Normalization



The medical report, sent through a *GET* request to the servlet, is divided into single words, that, if not recognized as *stopwords*, are normalized as follows:

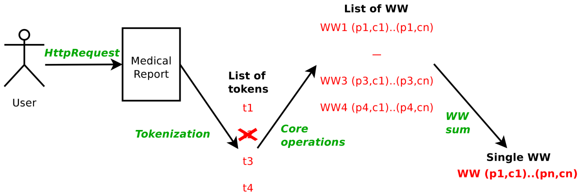
- everything is lower cased,
- accents, if any, are removed,
- tokens that matches one element of the PATTERN column of the DICTIONARY table, are replaced accordingly to the corresponding SUBSTITUTION.

Inverted indexes queries



For each token longer than three characters, accessing the core tables of the database, a *WeightedWord* object is created. These ones associate a string (the word) to a list of couples `(<code>,<weight>)`. The sum object is then calculated, and the list is ordered by weight.

Inverted indexes queries



User

HttpRequest

Medical Report

Tokenization

List of tokens

t1

~~t2~~

t3

t4

Core operations

List of WW

WW1 (p1,c1)..(p1,cn)

—

WW3 (p3,c1)..(p3,cn)

WW4 (p4,c1)..(p4,cn)

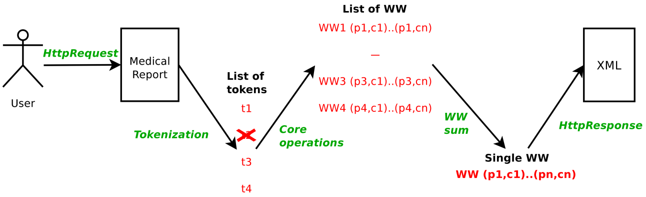
WW sum

Single WW

WW (p1,c1)..(pn,cn)

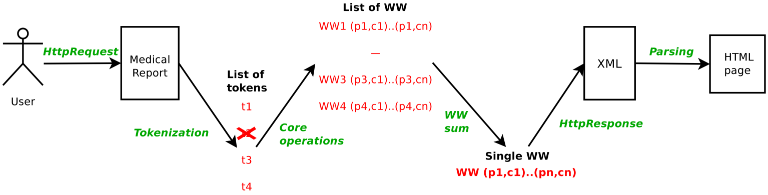
For each token longer than three characters, accessing the core tables of the database, a *WeightedWord* object is created. These ones associate a string (the word) to a list of couples (`<code>,<weight>`). The sum object is then calculated, and the list is ordered by weight.

Presentation of the results



The resultant *WeightedWord* object is sent to the client after being converted in an *XML* string. On the client side a *JavaScript* function parses it, showing a list of selectable codes with descriptions. Once the user has chosen one of the presented options, he is given the possibility of repeating the whole process, adding other diagnoses.

Presentation of the results



The resultant *WeightedWord* object is sent to the client after being converted in an *XML* string. On the client side a *JavaScript* function parses it, showing a list of selectable codes with descriptions. Once the user has chosen one of the presented options, he is given the possibility of repeating the whole process, adding other diagnoses.

Example of the XML string

```
<response>
  <item>
    <code>060</code>
    <description>Febbre gialla</description>
  </item>
  <item>
    <code>0600</code>
    <description>Febbre gialla silvestre</description>
  </item>
  ...
  <item>
    <code>0601</code>
    <description>Febbre gialla urbana</description>
  </item>
</response>
```

Focus on weight management

Idea

We based our choices on two simple observations:

Focus on weight management

Idea

We based our choices on two simple observations:

1. the less frequently a word appears in the specification, the more relevant it is likely to be,

Focus on weight management

Idea

We based our choices on two simple observations:

1. the less frequently a word appears in the specification, the more relevant it is likely to be,
2. the position a word occupies in the text is connected to its relevance, since we are dealing with search queries.

Focus on weight management

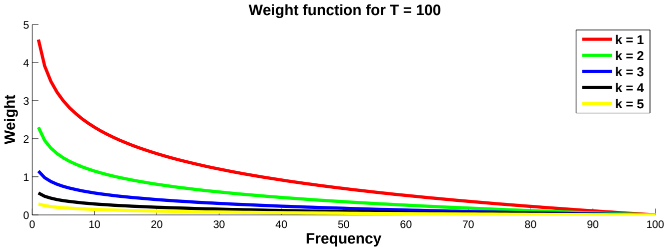
Idea

We based our choices on two simple observations:

1. the less frequently a word appears in the specification, the more relevant it is likely to be,
2. the position a word occupies in the text is connected to its relevance, since we are dealing with search queries.

After some trials with completely customized algorithms, we found an adaptation of the *inverse document frequency* approach (*idf*) very suitable for our purposes.
We obtained a good receipt by replacing the *term frequency* related factor (*tf*) with another metric, concerned with the position of the word in the string, as described into details below.

Focus on weight management



The formula we devised to calculate the weight of a term *i* is

$$w_i^k = \frac{1}{2^k} \log\left(\frac{T}{f_i}\right)$$

where

- k is the relative position of the word i in the typed string,
- f_i is the frequency of the word in the ICD-9-CM specification,
- T is the number of tuples of the **coding** table.

Addition of two *WeightedWord*

Example

By a logical viewpoint, a *WeightedWord* object has this structure: {gialla,{(060,3.45), (0600,3.45), (0601,3.45), (0609,3.45), (9793,3.45)}}

Addition of two *WeightedWord*

Example

By a logical viewpoint, a *WeightedWord* object has this structure:
{gialla,{(060,3.45), (0600,3.45), (0601,3.45), (0609,3.45), (9793,3.45)}}

The `plus()` static method of the *WeightedWord* class performs the logical union of two sets of *WeightedCode* objects, adding the weights associated to the same ICD-9-CM code.

Adding repeatedly, we obtain, from the medical report, a single *WeightedWord*, with the list of codes ordered by the `weight` field. If two codes have the same weight, the one with the shortest code string is preferred, since more generic.

History management

Every time a code and its description are chosen by the user, a new row is inserted in the HISTORY table. By clicking on the link Visualizza storico at the end of the main page, results of last successful searches can be viewed, organized as follows:

History management

Every time a code and its description are chosen by the user, a new row is inserted in the HISTORY table. By clicking on the link Visualizza storico at the end of the main page, results of last successful searches can be viewed, organized as follows:

- the textual medical report inserted,
- the selected code and its description,
- their corresponding date and time.

History management

Every time a code and its description are chosen by the user, a new row is inserted in the HISTORY table. By clicking on the link Visualizza storico at the end of the main page, results of last successful searches can be viewed, organized as follows:

- the textual medical report inserted,
- the selected code and its description,
- their corresponding date and time.

The DOCTORS and HOSPITALS tables are provided but not displayed yet, since we had no data to populate them.

Dictionary management

By clicking on Popola il dizionario delle abbreviazioni link it is possible to populate the column SUBSTITUTION of the DICTIONARY table, that contains acronyms and shortenings widely used in the medical context.

Dictionary management

By clicking on Popola il dizionario delle abbreviazioni link it is possible to populate the column SUBSTITUTION of the DICTIONARY table, that contains acronyms and shortenings widely used in the medical context.

The first column of that table contains the context in which the unknown word has been found, in which the pattern word is highlighted with a bold red font.

Dictionary management

By clicking on Popola il dizionario delle abbreviazioni link it is possible to populate the column SUBSTITUTION of the DICTIONARY table, that contains acronyms and shortenings widely used in the medical context.

The first column of that table contains the context in which the unknown word has been found, in which the pattern word is highlighted with a bold red font.

The collection of words to be substituted has been found working on a set of past diagnosis (from 2006 to 2011), selecting terms not in the Italian dictionary.

Some examples

Searching for *fibrosi* the results are:

- Fibrosi polmonare postinfiammatoria
- Fibrosi tubercolare del polmone
- Tiroidite fibrosa cronica
- Fibrosi cistica
- Fibrosi endomiocardica

Some examples

Searching for *fibrosi cistica* the results are:

- Fibrosi cistica
- Fibrosi cistica senza ileo da meconio
- Fibrosi cistica con ileo da meconio
- Fibrosi polmonare postinfiammatoria
- Fibrosi tubercolare del polmone

Some examples

Searching for *febbre* the results are:

- Febbre tifoide e paratifoide
- Febbre da morso di ratto
- Febbre gialla
- Febbre emorragica da artropodi
- Febbre ricorrente

Some examples

Searching for *febbre gialla* the results are:

- Febbre gialla
- Febbre gialla silvestre
- Febbre gialla urbana
- Febbre gialla non specificata
- Avvelenamento da vaccino contro la febbre gialla