Medical Terminology Server for the Hospital of Clinics of Paraguay using Apache Lucene and the UMLS Metathesaurus

Evelyn Aranda
National University of Asuncion, Polytechnic School, Asuncion, Paraguay,
evearandag@gmail.com

and

José Luis Vázquez Noguera
National University of Asuncion, Polytechnic School, Asuncion, Paraguay,
jlvazquez@pol.una.py

and

Cynthia Villalba
National University of Asuncion, Polytechnic School, Asuncion, Paraguay,
cvillalba@pol.una.py

Abstract
The current process of coding diagnoses in medical terminology standards, at the Hospital of Clinics of Paraguay, is carried out by physicians, usually residents. They search the medical terminology code in printed manuals or on their cell phones connected to internet. This search process takes a lot of time and has to be done during the medical consultation. This work proposes and evaluates a friendly medical terminology server, for Spanish speakers, through web services. The server uses Apache Lucene, as a search engine library, and the Metathesaurus of Unified Medical Language System (UMLS), as an information source. The results show that the process of coding diagnoses using the terminology server is 18 times faster than the current process. The user satisfaction degree is “Good” according to an adjective rating of the System Usability Scale (SUS). In addition, a comparison with a search engine of medical terminology called Metamorphosys shows that the implemented terminology server is quite competitive and it responds in a similar average time.

Keywords: Medical terminology, Hospital of Clinics of Paraguay, Apache Lucene, Metamorphosys.

1 Introduction
Diagnoses are coded daily at the Hospital of Clinics of Paraguay using medical terminologies coded in health standards. The Ministry of Public Health and Social Welfare of Paraguay established that medical diagnoses should be coded using the standard International Classification of Diseases Version 10 (ICD-10), since the World Health Organization recommends its use [1]. This standard provides statistical information on diseases or problems, which can be used for reporting and decision-making in the health area of a country or region [2].

At the end of the working day, and as extra work, physicians code the day’s medical diagnoses into health standards. The search for medical terminology code is done by consulting printed manuals or cell phones
connected to internet. This manual search process takes a lot of time and, for this reason, it cannot be carried out during the medical consultation.

This work focuses on solving the problem of search time for terminologies coded in standards, so we attempt to reduce it through a tool that responds with agility. In addition, we look to provide a user-friendly tool in Spanish language, which allows adding familiar or friendly terms, and associate them to coded medical terminologies. Thus, a physician will be able to search the coded terminologies using familiar terms.

Metamorphosys [3] is a searching tool of medical terminologies. Nevertheless, this tool is not adapted to the Hospital of Clinics of Paraguay due to its language and its search options. In addition, its use requires a license. For this reason, we implemented our own terminology server.

The server responds quickly and allows physicians to add new friendly terms that they can use in their searches. For the implementation of the server, we use a full-text search engine library, called Apache Lucene, which is specialized in persistence and quick recovery of information. UMLS Metathesaurus is used as the source of information about biomedical concepts, which is quite complete and provides a large number of controlled or standardized vocabularies. Security issues and authentication mechanisms are not covered in this work.

In this context, this work presents a novel application of existing open source technologies (Apache Lucene and Unified System of Medical Language) to develop an agile medical terminology server for Spanish speakers through web services. Furthermore, the test results show that it is a quite competitive terminology server.

The rest of the work is organized as follows: in Section 2 some preliminary definitions are presented; in Section 3, the background. Section 4 details the proposal for solution, which includes the design of the architecture. Section 5 explains the implementation. Section 6 presents the experiments carried out, the analysis and interpretation of results. Finally, the conclusions and future work are presented in Section 7.

2 Preliminary Definitions

In this section, we introduce the reader to important concepts related to the field of medical terminology server.

Terminology is defined as the complete set of words that have a specific meaning in a domain [4]. Specifically, the medical terminology expresses the complex concepts of medicine in precise terms and appropriate to their communication and expression needs [5]. In this context, the standards of medical terminology emerge to unify criteria and facilitate the exchange of information through a unique meaning of each term, accepted by the scientific community. These standards provide specific codes for clinical concepts such as diseases, list of problems, among others [5]. The task of physicians related to code clinical concepts by using standards of medical terminologies is commonly supported by medical terminology servers.

A medical terminology server is a software that maps the entered text (in natural language) to a list of complete, detailed, formal and standards-codified medical terminology [4]. Its main purpose is the representation of medical data as structured data, by coding it in health standards, so that it can be used in a database for management of information [7]. To do this, the server uses different types of terminologies: interface, reference and classification. The organization, structure and granularity of each terminology depend on its purpose. Figure 1 shows the organization of the terminology server used in this work. This model of terminology server was used by several works [6] - [4]. We describe each type of terminology as follows:

- **Interface terminology**: this represents the domain and local jargon, it is the language used by professionals in the medical record and has the advantage of using friendly and familiar medical terms [4].

- **Reference terminology**: they are terminologies designated to provide accurate representations of a domain of given knowledge. These terminologies are typically optimized to support storage and the recovery of clinical data from different ways [7]. An example of this kind of terminology is the Standard of “Systematic Nomenclature of Medicine - Clinical Terms” (SNOMED-CT) that also has a mapping to the classification terminology standard of ICD-10.

- **Classification terminology**: they allow to classify medical data such as diagnoses and diseases, in order to later perform an analysis. Some examples are the standards of “International Classification of Primary Care” (ICPC), “International Classification of Diseases Version 9” (ICD-9) and ICD-10.

These types of terminologies are independent from each other, but the terminology server maps them. The interface terminology allows mapping the natural language to the reference terminology. It also allows a rich synonymy, which enables physicians to represent clinical data using familiar words or phrases, but referring to the same concept of health.
In Figure 1, for example, we can see the terms “Atypical Pneumonia”, “Pneumonitis” and “Interstitial Pneumonia” within the interface terminology. These terms come from the natural language used by physicians. The interface terminology maps them to a single term from the reference terminology “Primary atypical pneumonia interstitial”.

Reference terminologies are often mapped to classification terminology. Figure 1 shows that the reference terminology term “Primary interstitial atypical pneumonia” has a mapping to the classification terminology in two standards: ICD-2 and ICD-10. Generally, in terminology servers, the common language is the starting point to a more structured (standardized) terminology such as those of reference and classification.

3 Background

In 1986, United States developed the Unified Medical Language System (UMLS). This open source system was created to facilitate the application of information technology systems in medicine and for software developers who would like to adapt it to their hospital systems.

UMLS is composed of three tools: a medical thesaurus called Metatessaurus, a semantic network and a search engine called Metamorphosys. The Metathesaurus is the integration of a thesaurus and a biomedical ontology. This tool integrates nearly 2,000,000 names, some 900,000 concepts from more than 60 families of biomedical vocabularies. The Metathesaurus is composed of several known terminologies standards such as “Systemic Nomenclature of Medicine - Clinical Terms” (SNOMED-CT) and the International Classification of Diseases Version 10 (ICD-10). The semantic network contains the relationships between the concepts that are handled in the Metathesaurus. Metamorphosys acts as a search engine and makes possible to navigate between the concepts of the semantic network and the metathesaurus. This search engine has four matching options for word search:

- Require match on lowest frequency word (option A): this option will only return concepts that contain the lowest frequency word of the search string. For example, a search on “Finger Protein” will only
consider candidates that contain “Finger” (the lower frequency word). Thus, “Protein, HN” would not be considered a candidate match.

- **Discard matches that only contain most frequent words (option B):** this option will only consider a candidate that contains either the lowest frequency word or multiple (higher frequency) words of the search string. For example, a search string like “Finger Protein Gene”, will consider candidates containing “Finger” or (“Protein” and “Gene”). Thus, “Zinc Finger Protein 51 Gene” would be considered a good match, “Gene Protein, Bacterial” a match, but not “Protein, HN”.

- **Basic matching (option C):** this option will find all concepts that have names matching any portion of the search string.

- **Require match on at least two words (option D):** this option only considers a candidate that contains at least two words of the search string.

Metamorphosys allows users to enter a text and returns a list of codified terminologies related to it. There is a proposal to allow interoperable access to UMLS knowledge using Fast Health Interoperability Resources [8].

The Italian Hospital of Buenos Aires (HIBA), in Argentina, has implemented its own terminology server [9] based on the structure of SNOMED-CT. Physicians can suggest to include familiar and friendly terms to the server. These suggestions are evaluated by specialists in terminology before incorporating them formally to the server. Initially, two million medical terms, identified in the hospital documents, were analyzed and then included in the terminology server [9].

Convergent Medical Terminology (CMT) is a source of definitions of medical concepts. The core is comprised by the following standards SNOMED-CT, LOINC, ICD-9 and ICD-10. CMT was developed in the United States by the Kaiser Permanente Company for its own use and to cover the needs in their electronic health records [10]. After 16 years of its distribution in the United States, Kaiser Permanente donated CMT to the International Health Terminology Standards Development Organisation (IHTSDO) [10]. CMT is free of charge to IHTSDO members [11].

The Veterans Health Administration of the States United (VHA) created a terminology of reference called Enterprise Reference Terminology (ERT) to promote the standardization of data. ERT contains terminology mapped to standards and for this purpose acquired licenses of SNOMED-CT, Current Procedural Terminology (CPT), UMLS and LOINC [12].

Apelon DTS is a distributed terminology system created in the United States. This tool provides a comprehensive terminology services for the creation and administration of local terminologies, ontologies and international standards [13]. Apelon key features include the support for HL7 FHIR terminology service standard, enterprise architecture, versioning, new metadata, enhanced web browsing and the editor. The version that is open source has limited functionalities [13]. Many healthcare information systems, such as the system of VHA [12] and Andalusia [14], use Apelon DTS as a medical terminology server. The free version of Apelon DTS was downloaded to analyze the feasibility of being used in the Hospital of Clinics of Paraguay. This version does not support SNOMED-CT and ICD-10, so that it is not suitable for use in the Hospital.

LexEVS is an open source terminology server, developed in the United States by the Mayo Clinic, together with the National Cancer Institute (NCI) and other collaborators. This server provides a model of common terminology and free access to a wide range of terminologies and mappings between standards that the NCI and their associates need. This tool is being used in other organizations like Stanford, Emory, the Medical Center of the Ohio State University, the University of Georgetown, the University of Washington, as well as commercial sellers such as IBM and General Electric Healthcare [15]. LexEVS works with several standards, but it is a complicated tool to use and to add new terms [16]. This tool requires non trivial tasks such as the configuration of the platform and the user training. LexEVS works with particular data types (objects of type LexEVS) [16].

Chile started the implementation of a terminology server called “Semantikos” that looks to standardize the language used in the field of health [6]. The terminology server aims to provide physicians with a terminology search engine rich in synonyms, acronyms and common or friendly terms used in the medical field. Semantikos uses the following standards: Nursing Outcomes Classification (NOC), Nursing Interventions Classification (NIC), the standard of the International North American Nursing Diagnosis Association (NANDA-I), SNOMED-CT (which includes official mappings to other classification terminologies such as ICD-9 and ICD-10), among others [6].

Ontoserver is a terminology server developed by Australia. It is based on the FHIR standard. It includes support for SNOMED CT, LOINC and Web Ontology Language (OWL), such as the Human Phenotype Ontology (HPO); a prefix-based search; a mechanism to facilitate keeping terminologies up to date; and enables data analytics. Its use is free in Australia [17].
Terminology Servers | Cost | Health Standards | Friendly terms | Language | Data language | Web Services
---|---|---|---|---|---|---
Metamorphosys | No cost | SNOMED-CT and CIE-9,10 | No | English | Several including Spanish | No, but offers a terminology query API
Apelon DTS | Paid and free version | SNOMED-CT, CIE-9, CIE-10, and others | Yes | English and Spanish | English and Spanish | Yes, SOAP type
LexEVS | No cost | SNOMED-CT, CIE-10, and others | Yes | English | English | No, but offers LexEVS API
HIBA | Paid | SNOMED-CT, CIE-9, CIE-10, and others | Yes | Spanish | Spanish | Yes
Ontoserver | Licence required for commercial use outside Australia | SNOMED CT, LOINC and OWL ontologies, FHIR | No | English | English | Include APIs

Table 1: Comparative summary of terminology servers

Several institutions in different countries tried to solve the problem of medical coding by using terminology servers with customized terminologies and adapted to each of country and institution. Table 1 presents a comparative summary of the main reviewed terminology servers.

Based on the tools found, it was decided to implement a terminology server since some of them are paid (Apelon DTS and HIBA), others just support English language (LexEVS and Ontoserver) or do not have the ability to add friendly terms (Metamorphosys and Ontoserver). Metamorphosys was analyzed in order to check the feasibility to be used as terminology search engine within the framework of this work, for this purpose preliminary searches were made. We identify the following problems in Metamorphosys:

- The response times are very different, even when we query the same text. The first search is very slow, reaching up to 2.7 minutes. This time is not acceptable for the user according to Jakob Nielsen [18]. The following searches of the same term are faster than the first one. This is because Metamorphosys stores information from the first search in cache memory, and this is used in the following searches.
- It was developed to support the installation and customization of the Metatesauro data, not precisely as terminology search engine.

The need of a terminology server to be accessed through external systems determines the architecture to be used. The next section presents in detail the proposed architecture.

4 Design

The design of the system contemplates two types of users:

- Physician: health professional in charge of suggesting friendly terms and consults the medical codings.
- Specialist in medical coding: professional with a broad knowledge about medical coding standards and in charge of validating the terms suggested by physicians.

For the architecture, we propose to use the three-layer client/server type, also known as a multilayer architecture. This architecture was chosen because the layers are maintained as independent modules with clear and specific objectives. The layers are presentation, business and data. The following subsections will explain in detail the architecture design.

4.1 Presentation layer

This layer is exposed on the client side. It is composed of interfaces that allow access to the business layer (see Subsection 4.2). The interfaces that integrate this layer are presented below.
• **Coding query interface**: used by the physicians, it allows the search for encodings in standards from friendly or natural terms.

• **Friendly terms suggestion interface**: used by the physicians, it allows to suggest the inclusion of a friendly term to the interface terminology associating it with a term of the reference terminology. This suggestion of inclusion and association must be validated by the specialist in medical coding. The association between the friendly term and the reference terminology determines the relationship between the interface terminology and the reference terminology. A friendly term may be associated to a single term of the reference terminology. However, a reference terminology term can have zero or several friendly terms associated with it. In Figure 1, for example, we can observe that the friendly terms “atypical pneumonia”, “pneumonitis” and “interstitial pneumonia”, of the interface terminology, refer to the same term “primary interstitial atypical pneumonia” from SNOMED-CT of the reference terminology.

• **Suggestion inclusion interface**: used by the specialist in medical coding, it allows him to agree the suggestion, in order to include the familiar term and associate it to a reference terminology standard, or reject it.

### 4.2 Business layer

This layer is composed of a set of web services and a text search engine library. This set of web services is consumed by the interfaces of the presentation layer and can also be consumed by other external systems. Following, we describe the set of web services.

• **Storage suggestion web service**: it is responsible for persisting in the database the suggestion of the association between the friendly term (to be included in the interface terminology) and the reference terminology term.

• **Recovery pending suggestions web service**: it retrieves the suggestions made by physicians that have not yet been validated from the data layer.

• **Web service for inclusion of suggestions**: it persists into the database the association between the suggested terms, previously validated by the specialist in medical coding.

• **Invalidation of pending suggestions web service**: it invalidates a suggestion in a validation pending state.

• **Coding search web service**: it receives a text and performs the search for medical codings, in standards of reference and classification, related to it.

The web services of inclusion of suggestions and coding search communicate with a text search tool. This tool is responsible for data recovering in an agile manner.

![Figure 2: Proposed architecture of the medical terminology server](image-url)
4.3 Data layer

This layer is made up of databases containing the medical terminologies encoded in standards and the suggestions made by the physicians.

4.4 General Proposed Architecture

The general architecture of the medical terminology server is presented in Figure 2. It is a three-layer client/server architecture. Each layer is composed as follows:

- **Presentation layer:**
  1) Coding query interface.
  2) Friendly terms suggestion interface.
  3) Suggestion inclusion interface.

- **Business layer:**
  1) Storage suggestions web service.
  2) Recover pending suggestions web service.
  3) Web service for inclusion of suggestions.
  4) Invalidation of pending suggestions web service.
  5) Coding search web service.
  6) Text search tool.

- **Data layer:**
  1) Database.

The Coding Query Interface (1) receives as an input the text entered by the physician and communicates with the Coding Search Web Service (5). This web service uses the Text Search Tool (6) to recover quickly the information of the database.

The Friendly Terms Suggestion Interface (2) receives two terms entered by the physician: the friendly term and the term of the reference terminology to which associate with the friendly term. The interface (2) of the presentation layer communicates with the Storage Suggestions Web Service (1) of the business layer. This web service communicates with the data layer to persist the suggestion made by the physician.

The specialist in medical coding is responsible of validate the suggestions made by the physicians. The specialist manages the Suggestion Inclusion Interface (3) that allows him to perform three actions:

- Recover the suggestions made by physicians that have not been validated yet. The Suggestion Inclusion Interface (3) communicates with the Recover Pending Suggestions Web Service (2) that queries the database to retrieve the suggestions that are pending to validate and to expose them to the specialist in medical coding in charge of the validation.

- Invalidate pending suggestions in case the suggestions are not correct. The specialist in medical coding chooses a suggestion that considers invalid. The Suggestion Inclusion Interface (3) communicates with the Invalidation of Pending Suggestions Web Service (4) that communicates with the data layer to modify the state of suggestion.

- Include the correct suggestions into the database. The specialist chooses the suggestion pending of validation. If he considers that the suggestion is correct, interface (3) communicates with Web Service for Inclusion of Suggestions (3). This web service uses the Text Search Tool (6) to quickly persist the information in the database.

In this way we can observe the interaction between the different layers of the multilayer client/server architecture. In the following section, the implementation of the components of each layer is presented.

5 Implementation

This section discusses the tools used in the terminology server and the implementation detail of each of the layers.
5.1 Used tools

Implementation requires the search for tools that cover the aspects included in the design. The text search tool used is the Apache Lucene text search engine library. The data used as a source of information for the terminology server is extracted from the UMLS Metathesaurus (mentioned in Section 3).

- **UMLS Metathesaurus**: is a fairly comprehensive source of information on biomedical concepts. It provides a large number of controlled or standard vocabularies, making it an almost universal reference of concepts. Other sources of information such as CMT and ERT (mentioned above) do not have information in Spanish, so they were not selected. The installation of the Metathesaurus is done using the UMLS MetamorphoSys tool. This tool allows to store in MySQL or Oracle databases, the information contained in the Metathesaurus and has a search engine for accessing the installed data [3]. The speed of response time for data recovery is a determining factor and therefore a tool that fulfills this requirement is sought. The selected tool is fast in response time. It will be detailed below.

- **Apache Lucene**: is a high performance, scalable text search engine library. It focuses on indexing and full-text search on any data that can be converted to text [19]. Apache Lucene is a library, but it is often used as a search engine. Its core consists of three interrelated concepts used in search engines. These concepts are: indexing, analysis and search [20]. Egothor and Xapian are other information retrieval libraries with quite similar features. However, we selected Apache Lucene because of the programming language in which it is implemented. The knowledge of this programming language makes it easier to modify it if necessary. Apache Lucene is implemented in Java, the language that was chosen for the development of this work.

5.2 Layers implementation

The three layers of architecture interact with each other through various elements. The interaction between them is shown in Figure 3. The interfaces of the layer presentation communicate with the web services of the layer of business through HTTP requests using the technical AJAX (JavaScript and XML asynchronous). HTTP requests are messages sent by a client to initiate an action on the server [21]. AJAX is an asynchronous technique that allows data to be exchanged with the server and pages to be updated on the client side without the client being aware of it. The interfaces are not waiting for a response from the server [22].

The web services of the business layer communicate with the relational database of the data layer through a JDBC driver that allows the connection to the database. The business layer also communicates with the data layer through the text search engine library, Apache Lucene. This library communicates directly with a data directory from which it retrieves and stores the information.

The implementation of each of the layers of the architecture is detailed below:

- **Presentation layer**: the interfaces were implemented using JavaServer Pages (JSP), which is a technology used to create dynamic web pages. JSP files are compiled as any Java class in an application container and therefore it is a server-side technology. The interfaces use HTML tags and allow you to include Java code in your pages [23].

   *Twitter Bootstrap* (open source framework) in its version 3.3.7 was used for issues of design and functionality of the interfaces. It contains HTML and CSS based design templates, plus additional
JavaScript extensions. The choice is also due to the fact that the framework is supported in most existing browsers. In addition, it includes by default responsive functionality from version 2 onwards, fitting perfectly into a mobile environment [24].

Friendly terms suggestion interface is shown in Figure 4. This interface allows to suggest a relationship between a friendly term and a term of the reference terminology, through the entry of data in the fields Suggested Term (Término sugerido) and Formal Term (Término formal), respectively.

Suggestion inclusion interface is shown in Figure 5. This interface allows the coding specialist to choose a suggestion to be validated and register it or to invalidate a suggestion in case it is not correct. When selecting a pending suggestion using the combobox, the fields: Suggested term (Término sugerido) and Associated to the formal term (Asociado al término formal) are preloaded with the data of the chosen suggestion.

The specialist can edit these two fields in case they require a correction. This user has two options to follow: the invalidation of the suggestion or the search for the reference terminology to which to associate the friendly term.

![Sugerencia de nuevos términos](image)

Figure 4: Friendly terms suggestion interface
The **Find Terminology** (Buscar Terminología) button allows the search for terminologies coded in standards associated with the term entered in the **Associated to the formal term field** (Asociado al término formal). This search returns a list of encodings in standards that are preloaded in the combobox of the second step. The specialist selects the coding of the reference terminology that best fits and presses the **OK** (Aceptar) button. This button ensures that the relationship between the friendly term in the interface terminology and the chosen term in the reference terminology is maintained. The coding query interface is shown in Figure 6. This interface allows the doctor to enter the search text in the **Term to consult** (Término a consultar) field, returning a list of terminologies encoded in reference and classification standards, associated with the term consulted. The pattern returned by the search engine is the following: description of the terminology, the name of the standard and the coding.

In Figure 4 the suggested friendly term “papera” was associated with a single term from the reference terminology (“parotiditis” from the SNOMED-CT standard with code 36989005). This term from the reference terminology (in the SNOMED-CT standard) is associated with several terms from the classification terminology (in the CIE-10 standard). In Figure 6 it can be seen that when asked about the friendly term “papera”, the terminology server responds with the terms of the reference terminology (SNOMED-CT) and the classification terminology (ICD-10).

- **Business layer**: it is composed of web services of the type Representative State Transfer (REST) using Java. This service is consumed by the interfaces of the presentation layer through AJAX requests. The design principles of REST type web services are as follows 25:
  - It uses HTTP methods explicitly to request actions on the server.
  - It does not maintain any status about the client.
  - It exposes URIs in the form of directories.
  - It transfers XML, JavaScript Object Notation (JSON), or both.

These principles determine the choice of the REST type for the implementation of services. The following describes the implementation of the web services (mentioned in section 4):

- **Storage suggestion web service**: receives the friendly term and the reference terminology term to which you want to associate, through the AJAX request made by the friendly terms suggestion interface. This web service communicates with the relational database and inserts the suggestion with PENDING status in a table where all the suggestions made are recorded.
Recover pending suggestion web service: communicates with the database relational to retrieve all the suggestions with pending status. This service is consumed by the suggestion inclusion interface so that the suggestions can be validated and registered by the coding specialist.

Web service for inclusion of suggestions: receives the friendly term, the term of the reference terminology itself and the codes for the latter. The service is used to register a suggestion with PENDING status. The service communicates with the relational database for the update of the suggestion status to the FINISHED status. Apache Lucene indexes the suggestion data and stores it in a data directory created by it.

Invalidation of pending suggestions web service: invalidates a pending suggestion with an id identifier (received as parameter), updating the suggestion to the CANCELED status. This service is consumed by the Suggestion inclusion interface.

Coding search web service: receives a search text and returns a list of terminologies coded with reference and classification standards. The service uses the text search engine library for the recovery of information contained in data repositories. In this process of storing and retrieving information, Apache Lucene uses components to analyze the words, called “Analyzers”. By default, Apache Lucene does not have parsers for accented words, which is why in this work an analyzer was implemented to support the search for this type of words.

Data layer: consists of a relational database (implemented in MySQL version 5.6) and a data repository (or non-relational database) created by Apache Lucene. The relational database is made up of the data from the Metosaurus and the physician’s pending suggestions, as shown in Figure 7. The data repository (not relational), is formed by the data that Apache Lucene indexes for later searching. Apache Lucene takes the data to be indexed from the relational database, limiting the data to the type “Problems or Diseases” of the SNOMED-CT and CIE10 standards. In this work, the data universe was limited to these standards, because the Hospital de Clínicas del Paraguay codes in the ICD-10 standard and limited to SNOMED-CT as reference terminology because it has ICD-10 mapping. Thus,
there is a standard for reference terminology and another for classification terminology. In addition, these two standards are the most widely used worldwide [4].

The suggestions made by the specialist, once Apache Lucene indexes the data, are stored in the non relational database.

The web services mentioned in the Business Layer are publicly available for testing (according to the format of each web service specified above) on a server with IP 190.52.180.236 on port 8585. An example of a service call (using a web service client), is as follows: `http://190.52.180.236:8585/termserver/sft/search?diagnostico=papera`

Figure 8 shows the example web service query and the corresponding response, using the Postman tool [26] developed to test Web services.

### 6 Experiments, Analysis and Results Interpretations

This section presents the different experiments carried out to evaluate the terminology server implemented. The experiments were performed in a local machine, then there was no delay of internet connection. The
computational environment over which it was tested the terminology server has the following characteristics:

- Processor: Intel core duo 2.7 GHz.
- Amount of RAM memory: 4 GB.
- Hard disk capacity: 500 GB.

The performed experiments are organized as follows:

- In the first part (Subsection 6.1), the response time of the terminology server is measured and compared with the response time of the current terminology search process at the Hospital of Clinics.
- In the second part (Subsection 6.2), we tested the search for coding terminologies by using friendly terms. This experiment checks the objective of establishing searches for coding terminologies by using friendly terms.
- In the third part (Subsection 6.3), we compare the response time of the implemented server with the response time of another tool (Metamorphosys). In this experiment, we want to evaluate the agility, related to the response time, of the implemented server.

In addition, Subsection 6.4 presents an evaluation to know the degree of satisfaction of users by using a questionnaire of usability called “System Usability Scale” (SUS) [27].

### 6.1 Experiment 1: Assessment of the response time of the terminology server.

The objective of this experiment is to measure the average time to search for diagnoses by using the terminology server and compare it with the average time to search by using the current process at the Hospital of Clinics by employing the physicians’ cell phones connected to internet.

A total of thirteen resident physicians from the Pediatrics area of the Hospital of Clinics performed the diagnoses’ searches by using the terminology server, in order to obtain the terminologies codified in standards. Table 2 shows the number of physicians that carried out the tests and the amount of diagnoses’ searches performed. The total of diagnoses’ searches was 64, since one physician made 4 diagnoses’ searches and the other ones (12) performed 5 diagnoses’ searches each. The decision to perform 5 diagnoses’ searches by each physician was due to a time limitation of physician. The terms used in the searches were those commonly used by physicians in the medical records of their specific area. The log file collected the response times of the server.

<table>
<thead>
<tr>
<th>Number of physicians</th>
<th>Number of diagnoses</th>
<th>Total diagnoses’ searches</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>5</td>
<td>60</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 2: Data from Experiment 1

The average code search time of the 64 diagnoses by employing the physicians’ cell phones (connected to internet), like the current process at the hospital, was 18 seconds and by using the terminology server was 1 second. It is necessary to denote that the searches using the implemented server were performed in a local machine, then there was no delay of internet connection. Therefore, the search time of the current process may vary depending on the speed of the internet and the number of users connected to it. The experiment was carried out under the assumption that if the medical service implements the terminology server, it would have its own local database. Under these criteria the terminology server turns out to be 18 times faster than the current process. The implemented terminology server, with a local database, reduces the search time of the current process.

### 6.2 Experiment 2: Search of coded terminologies using friendly terms.

This is a simple experiment that only tests the suggestion functionality mentioned in the Section 4. The terminology server allows physicians suggest to include familiar or friendly terms and to associate it with the coded terminologies. The physicians could use these friendly terms to perform terminologies searches coded in standards.

Three resident physicians made a suggestion of three friendly terms each, by using the friendly terms suggestion interface. The suggestions were validated by the developer of the application who acts as a
specialist in medical coding due to the lack of one at the Hospital of Clinics. The friendly terms validated and included into the server were used to perform searches of terminologies coded in standards. According to the physicians, the three searches made using these friendly terms responded with the appropriate coding terminologies for each case. In this way it is concluded that the searches using the friendly terms were successful.

Table 3 shows the suggestions made by physicians. The friendly term “papera” was suggested to be associated with the reference terminology “Parotitis”. This suggestion was validated by the specialist in codification. In this way the terminology server knows the new friendly term “papera” and relates it with the term “Parotiditis” used in medical coding standards. The search for the term “papera” in the terminology server now throws terminologies coded in standards as displayed in the last column.

<table>
<thead>
<tr>
<th>Friendly suggested term</th>
<th>Associated reference terminology</th>
<th>Successful search</th>
<th>Server response (shows only one result as an example)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Papera</td>
<td>Parotiditis</td>
<td>Yes</td>
<td>Parotiditis, sin complicaciones ICD10 - B26.9</td>
</tr>
<tr>
<td>Diarrea</td>
<td>Gastroenteritis</td>
<td>Yes</td>
<td>Colitis y gastroenteritis no infecciosas, no especificadas ICD10 - K52.9</td>
</tr>
<tr>
<td>HTA</td>
<td>Hipertension arterial</td>
<td>Yes</td>
<td>hipertensión arterial secundaria benigna SCTSPA - 194785008</td>
</tr>
</tbody>
</table>

Table 3: Search of terminologies coded in standards using friendly terms

6.3 Experiment 3: Comparison of the implemented server response time with the Metamorphosys search engine.

In this experiment the response times of the implemented terminology server are compared with those of Metamorphosys (mentioned in Section 3). The same terms searched in Experiment 1 with the implemented terminology server were also searched with Metamorphosys, using each one of its four options. It is worth mentioning that Metamorphosys does not support the search of words with accent, therefore we implemented a mechanism that allows physicians to make this kind of search.

Figures 9, 10, 11 and 12 show the response times for each option of Metamorphosys and for the implemented server. The atypical values were eliminated from these charts in order to better visualize the average
Figure 10: Response times of Metamorphosys Option B versus Implemented Server

Figure 11: Response times of Metamorphosys Option C versus Implemented Server

<table>
<thead>
<tr>
<th></th>
<th>option A</th>
<th>option B</th>
<th>option C</th>
<th>option D</th>
<th>Implemented Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average response times (in milliseconds)</td>
<td>583</td>
<td>499</td>
<td>523</td>
<td>1006</td>
<td>902</td>
</tr>
</tbody>
</table>

Table 4: Average response time
Figure 12: Response times of Metamorphosys Option D versus Implemented Server

Figure 13: Metamorphosys and Implemented Server including atypical values
1. I think that I would like to use this tool frequently.
2. I found the tool unnecessarily complex.
3. I thought the tool was easy to use.
4. I think that I would need the support of a technical person to be able to use this tool.
5. I found the various functions in this tool well integrated.
6. I thought there was too much inconsistency in this tool.
7. I would imagine that most people would learn to use this tool very quickly.
8. I found the tool very awkward to use.
9. I felt very confident using the tool.
10. I needed to learn a lot of things before I could get going with this tool.

Table 5: SUS questionnaire

<table>
<thead>
<tr>
<th>Strongly disagree</th>
<th>Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Strongly agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. I think that I would like to use this tool frequently.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. I found the tool unnecessarily complex.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. I thought the tool was easy to use.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. I think that I would need the support of a technical person to be able to use this tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. I found the various functions in this tool well integrated.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. I thought there was too much inconsistency in this tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. I would imagine that most people would learn to use this tool very quickly.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. I found the tool very awkward to use.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. I felt very confident using the tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. I needed to learn a lot of things before I could get going with this tool.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

results. Despite this, the atypical values are important because they influence on the average and provide information about the general behavior of results. Therefore, they are included in the later scatter chart (Figure 13).

The scatter charts show that both Metamorphosys and the implemented server have similar response times. Slight differences are observed in Figures 9, 10 and 11. Table 4 presents the average response time of Metamorphosys and of the implemented server excluding atypical values. It is observed that both respond in approximately 1 second or less. Based on these results, it can be concluded that both Metamorphosys as the implemented server have practically the same response time.

As we mentioned earlier, some atypical values were identified. Figure 13 shows atypical values for options C and D of Metamorphosys with response times of up to 25000 and 32000 ms, respectively. The implemented server reaches a maximum atypical value and maximum response time of 6100 ms. Metamorphosys presents atypical values when we search a term for the first time. The term is stored in the cache memory so that later searches are faster by consulting this memory and not the disk. For example, when the term “Diarrea crónica” is searched for the first time, Metamorphosys responds in 7 seconds. In the following searches of the same term, Metamorphosys responds in approximately 0.4 seconds. This memory caching process depends on the database configuration that is done during the Metamorphosys installation. Also, the amount of available memory affects the Metamorphosys performance. Metamorphosys requires 3.5 GB of allocated memory for its correct operation. When the amount of information to return is very big, it becomes slower and even, in some cases, stop responding by have run out of available memory, after having consumed approximately 3 GB of memory. If Metamorphosys begins to reach the limit of its available memory, it takes much more time to respond, reaching even at 32 seconds approximately.

In the mentioned experiments it was possible to evaluate the terminology server performance in terms of response time and the functionality of suggest friendly terms. Another interesting metric to assess is the degree of user satisfaction. In the next section, we present the results about the physician satisfaction level using the terminology server.

6.4 Evaluation of user satisfaction

The physicians completed the SUS questionnaire after performing searches of medical terminologies coded in health standards. The questionnaire is shown in Table 5. It consists of ten questions that help to assess the user satisfaction degree. This questionnaire was chosen because it is the questionnaire most used to measure usability perceptions, it is independent of the technology, that is, it can be used to evaluate
Table 6: SUS score

<table>
<thead>
<tr>
<th>Item</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>Total</th>
<th>Total * 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Punctuation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6 shows the calculation of the SUS questionnaire punctuation, where the value of each item is given as follows:

- For even items: $5 - \text{scale position}$. That is, subtract its scale position from five.
- For odd items: $\text{scale position} - 1$. That is, subtract one from its scale position.

Figure 14 shows the SUS scores of the questionnaires completed by each physician. The general SUS score obtained was 83.46, that means that the implemented terminology server has a “Good” user satisfaction degree, according to Bangor et al. [27].

7 Conclusion and Future Work

This work presents the design, implementation and evaluation of a medical terminology server for the Hospital of Clinics of Paraguay.

The server receives as input friendly or familiar terms in Spanish, entered by physicians in natural language, and quickly returns a list of medical terminology codified in health standards. The architecture consists of a three-layer client/server. The implementation of the server is based on web services so that they can be consumed by the interfaces implemented in this work and other external systems. Apache Lucene gives the required agility to the solution. This search engine library specializes in persistence and quick recovery of information thanks to the indexing technique that uses and to its full-text search approach. UMLS Metathesaurus is used as the source of information about biomedical concepts. It is quite complete and provides a large number of controlled or standardized vocabularies.

The evaluation results show that the implemented server responds on average in one second, which is up to 18 times faster than the current manual search process carry out on the physicians’ cell phone connected to internet. In addition, physicians have successfully tested the functionality of searching medical terminology codes given friendly or familiar terms entered in natural language. These terms should be previously associated with the corresponding medical terminology codes by the specialist in health standards. The user satisfaction degree is “Good” according to the adjective rating of the System Usability Scale (SUS) proposed by Bangor et. al. [27].

Furthermore, the implemented terminology server shows results quite competitive compared with Meta-morphosys. For both, the response time is one second on average approximately. Metamorphosys, however,
presents very high typical values in some cases, e.g. first time term search. Its performance is affected by the amount of available memory. When it begins to reach the limit of his available memory, it takes more time to respond, reaching even at 32 seconds. The terminology server has the advantage of taking Spanish accented words into account, while Metamorphosys does not recognize them.

As future works, there are issues related to security, authentication and integration of the terminology server with information health systems.

References


