TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Master's Thesis

UMLS for Information Extraction

Institute of Software Technology & Interactive Systems

Information Engineering Group

Vienna University of Technology

Supervised by

Ao.Univ.Prof. Mag. Dr. Silvia Miksch

Mag. Dr. Katharina Kaiser

By

Michael Kohler

Westbahnstrasse 46/7

1070 Vienna

Vienna, Mai 07, 2007

Contents

Abstract

The enormous growth of the world wide flood of information makes it more and more important to use effective tools to extract and condense key information. There are ongoing researches in the branch of Natural Language Processing (NLP). Information Extraction (IE) is a section of NLP and is used to extract information from text to fill a database. However, there are limitations in the use of IE. The IE systems need to be specialised on a specific domain and therefore they are only able to handle text from an indicated domain. IE systems are consisting of several components, one of the important components may be composed of terminologies, ontologies, and vocabularies.

The UMLS combines a huge variety of source vocabularies, terminologies, and ontologies to the SPECIALIST lexicon, the Metathesaurus, and the Semantic Network. The UMLS is a gigantic knowledge base, which covers numerous themes in medicine.

Due the large size of umls, it is difficult to extract information. Also matching concepts to phrases is not an easy task. With the help of MMTx the matching problem can be outsourced.

To break down the complex data structure of UMLS and MMTx, a more simple and easy accessible data structure was introduced, which is part of the UMLSint package. The UMLSint package was developed to simplify the access to the UMLS data, to extract the attributes, which are of interest, and to analyse the input data to find the referring concepts in the knowledge base. The UMLSint package gets as an input a sentence of medical text and returns attributes of interest from the UMLS in accordance to questioned phrase. The information consists of factual knowledge from the Metathesaurus and information generated by the MetaMap Transfer (MMTx) tool. The MMTx tool is used to create logical elements and gather information about the lexical and morphological structure.

For each logical element various information is now accessible, such as semantic type, term type, Part-Of-Speech tag, Metathesaurus concept ID, and many more. This information can be used for both NLP and IE systems for further analysis of the text.

The subject of this thesis is to enable IE systems, which process medical text, an easier access to the knowledge base named Unified Medical Language System (UMLS).

Zusammenfassung

Das enorme Wachstum in der weltweiten Flut von Informationen führt zwangläufig dazu, wirksame Werkzeuge zu entwickeln um die Informationen zu filtern und zu komprimieren. Im Bereich von Natural Language Processing (NLP) werden andauernde Forschungen durchgeführt. Information Extraction (IE) ist ein Teilbereich von NLP und wird dazu verwendet, Informationen aus Texten zu extrahieren um damit eine Datenbank zu füllen. In der Verwendung von IE, gibt es jedoch Einschränkungen. IE Systeme müssen jedoch auf eine bestimmte Domäne spezialisiert werden und nur dann sind sie in der Lage Texte von einer dieser Domäne zu verarbeiten. IE Systeme setzen sich aus mehreren Komponenten zusammen, eine der wichtigsten Bestandteile kann aus Terminologien, Ontologien und Vokabularen bestehen.

Das UMLS System besteht aus einer Vielzahl von Wörterbüchern, Thesauri, Terminologien und Ontologien, die Mithilfe des SPECIALIST Lexicon, dem Metathesaurus und dem Semantic Network dargestellt werden. Das UMLS ist eine gigantische Wissensbasis, welches eine Vielzahl von medizinischen Themen umfasst.

Durch die enorme Größe von UMLS ist es schwierig, Information herauszubekommen. Auch die korrekte Zuweisung von Phrasen zu Konzepten ist keine einfache Aufgabe. Mit der Hilfe von MetaMap Transfer (MMTx) kann dieses Zuweisungsproblem ausgelagert werden.

Um die komplexe Datenstruktur von UMLS und MMTx aufzuschlüsseln, wurde eine einfachere und leichter handhabbare Datenstruktur eingeführt, welche Teil des UMLSint package ist. Das UMLSint package wurde entwickelt, um den Zugang zu den UMLS-Daten zu vereinfachen, die Attribute, welche von Interesse sind zu extrahieren, und die Eingabedaten zu analysieren, um die betreffenden Konzepte in der Knowledge Base zu finden. Das UMLSint-Paket erhält als Eingabe einen Satz eines medizinischen Textes und liefert die Attribute von Interesse von UMLS zurück. Die zurück gelieferten Informationen, bestehen aus dem Faktenwissen des Metathesaurus und aus den generierten Informationen des MMTx Werkzeuges. Dieses MMTx Werkzeug wird dazu benutzt, um logische Einheiten zu erzeugen und Informationen über die lexikalische und morphologische Struktur zu erzeugen.

Für jede logische Einheit werden verschiedene Informationen, wie semantische Art, Begriffsart, Wortart, Metathesaurus Konzept ID und vieles mehr zurückgeliefert. Diese Informationen können sowohl für die Verarbeitung der natürlichen Sprache, wie auch für IE Systeme zur weiteren Analyse des Textes verwendet werden.

Das Thema dieser Arbeit ist es, IE Systemen, welche medizinische Texte verarbeiten, einen leichteren Zugang zur Knowledge Base zu verschaffen, welche hier UMLS System darstellen.

1 Introduction

"We drown in information but hunger for knowledge."

John Naisbitt (*1930), American Forecaster

The number of published documents increases continually in an ever faster cycle. There is an incalculably big amount of texts and information for certain topics and concepts. And with the help of the World Wide Web even more written texts are coming available. The increase of articles is enormous and of course scientific articles are not spared within this flood of words.

Some numbers to show the annual increase:

- At the Vienna University of Technology publications increase by 10,000 each year [1]
- TEMA Technic and Managment Database: documents increase by 120,000 each year [2]
- ZDE Electrotechnics and Electronics Database: documents increase by 50,000 each year [2]
- MEDITEC Medizintechnic Database: documents increase by 10,000 each year [2]

There are approximately 100,000 magazines and 10 millions of articles published each year. These numbers are from the year 2000 and it is obvious to imagine as in Figure 1-1 that we are not able to manage all the information without the help of new technologies [3].

To be able to process this enormous information flood, new methods were developed, called Natural Language Processing (NLP). These methods support the automatic analysis and refurbishing of texts. One task is Information Retrieval (IR), which serves to search for documents containing specific information within a collection of documents. A specialization of this subject is called Information Extraction (IE). The tasks of IE are the extraction of information and relations from „machine-readable“ documents and populate a database with the received information. This branch was developed during the first Message Understanding Conference (MUC) in 1987 in which it first was considered only a simplified problem of NLP and then later developed into a discipline of its own, since the topic and problem solutions were more complex and more different than assumed [4].

A popular definition from Yangarber is that IE is "an emerging NLP technology whose function is to process unstructured, natural language text, to locate specific pieces of information, or facts in the text, and to use these facts to fill a database." [5]

There are many application areas for IE systems as to read out newspaper articles, scientific reports, and it is possible to process nearly everything what is getting written. Most IE systems can only process text from one specific topic. In this thesis we work with medical documents, so called clinical practice guidelines (CPGs), "systematically developed statements to assist practitioners and patient decisions about appropriate health care for specific circumstances" [6].

Figure 1-1 Information Overflow [7]

There are several CPGs for a lot of diseases; most diseases do have at least four to five different guidelines. For example, there are 2,038 summaries of CPGs on the National Guideline Clearinghouse (NGC) [8]. A goal of IE is to analyze the CPGs from one disease and obtain information from them. With the gathered information, it is now possible to compare the guidelines and even enhance them.

To be able to analyze those articles, IE systems do need some kind of specialized vocabulary. Some of them use ontologies. The term *ontology* was first used in philosophy, to describe the subject of existence [9]. In computer science "*ontology"* means something different. Gruber uses the term to mean a "specification of a conceptualization" [10]. And furthermore he says: "An ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents" [10].

Examples for specialized vocabularies and thesauri are:

- Medical Subject Headings classification (MeSH): It *"is a controlled vocabulary produced by the National Library of Medicine and used for indexing, cataloguing, and searching for biomedical and health-related information and documents*" [11]
- Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT) [12]: It is a hierarchical classification system, with a coding system.
- International Classification of Diseases (ICD): "*It is used to classify diseases and other health problems recorded on many types of health and vital records including death certificates and hospital records"* [13].

A far more ambitious approach is the Unified Medical Language System (UMLS) [14] established by US National Library of Medicine (NLM). It is an attempt to classify and define the language of the medicine and health supply, so computer systems have a basis in which they are able to analyze medical texts.

The main task in my thesis is now to find out how the information provided by the UMLS can be used by an IE system, which can process CPGs to extract relevant information into a data-base. UMLS consists of Semantic Network, Metathesaurus, Specialist Lexicon and Lexical Tools. With the help of UMLS I want to improve IE systems processing CPGs using semantic.

In Chapter 2 are more detailed information about the UMLS and its knowledge sources and used tools is given. In Chapter 3 there are detailed explanations about Information Extraction and in Chapter 4 the Java implementations for the interface to UMLS are documented, and explained how and why I accomplished it. Chapter 5 contains the case study to clarify the methods used in the UMLSint package and explain their way of function with the help of some sample sentences. Chapter 6 consists of the summary and future work and Chapter 7 contains the conclusion.

2 Unified Medical Language System (UMLS)

In medicine there are innumerable databases, dictionaries, specialized vocabularies and ontologies available, which deal with illnesses, therapies, diagnoses, and so on. Those systems try to store medical knowledge in a well structured format. How the knowledge is stored, depends on the goal of the developer.

- The clinical focus of ICD is an international comparability of mortality statistics [13].
- The clinical focus of SNOMED CT is to advance excellence in patient care [12].
- The clinical focus of MeSH is indexing documents containing information about healthcare and biomedicine [11].

That are only a few examples of common vocabularies, but it is easy to see that they have different goals. The problem is, that they are not useable together, because they all use different methods how to store and organize their information.

The NLM initiated UMLS in 1989. It is an attempt to fill the gap and to connect the individual vocabularies among each other to attain an almost complete picture of the medical knowledge.

> "*The purpose of NLM's Unified Medical Language System (UMLS®) is to facilitate the development of computer systems that behave as if they "understand" the meaning of the language of biomedicine and health.*" [14]

UMLS consists of three knowledge sources, which are the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon. The Metathesaurus stores the concepts, the Semantic Network holds all the categories and relations for the concepts and the SPECIALIST Lexicon is used to generate the indexes to the Metathesaurus.

One of the most important issues is to conserve every integrated vocabulary in the UMLS.

> *"The UMLS approach assumes continuing diversity in the formats and vocabularies of different information sources and in the language employed by different elements of the biomedical community. It is not an attempt to build a single standard biomedical vocabulary."* [15]

Therefore, every concept, every relation, every information of the different vocabularies are preserved and linked together with each other as shown in Figure 2-1.

Different concepts and systems are represented in the uppermost level of Figure 2-1, how to link a certain illness and their symptoms with each other. As one can see, the different concepts (yellow, blue, green) have different dependencies; however, it is the same subject. UMLS tries to merge these concepts with each other without losing information. As one can see in Figure 2-1 a single big tree arises, which contains several paths. By merging this information additional knowledge was created indirectly. For example, the node E has in UMLS A, C and B as parent node, and one also can reach node G from the node C now, which was not possible with the previous tree structure.

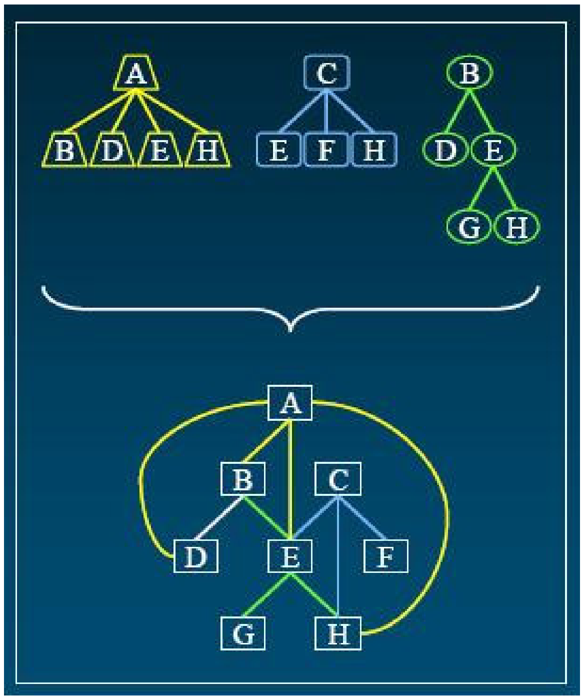


Figure 2-1 shows merging structures from different source vocabularies together. The different colours, yellow, blue and green stand for different concepts, and the resulting tree stands for an example how UMLS merges the information together [16].

2.1 Knowledge Sources

UMLS consists of three knowledge sources, which fulfil different functions. They are explained in the following chapters.

2.1.1 Metathesaurus

The Metathesaurus is a very large vocabulary database, which contains information about biomedical and health care. It now contains more than 1 million concepts and 5 million unique concept names from more than 100 different source vocabularies. Each concept is linked to the other two knowledge sources, which provide additional information.

> *"The Metathesaurus is organized by concept or meaning. In essence, its purpose is to link alternative names and views of the same concept together and to identify useful relationships between different concepts."* [14]

There are a few rules, how to merge the different source vocabularies together.

- If two different source vocabularies share one name for different concepts, they are both stored in the Metathesaurus and both are displayed, once found. They still represent the meaning of the original source vocabularies.
- If identical concepts appear in different hierarchical contexts, the Metathesaurus includes all the hierarchies.
- Different views from different source vocabularies for relations between concepts are also included into the Metathesaurus.

> *"In other words, the Metathesaurus does not represent a comprehensive NLM-authored ontology of biomedicine or a single consistent view of the world (except*

at the high level of the semantic types assigned to all its concepts). The Metathesaurus preserves the many views of the world present in its source vocabularies because these different views may be useful for different tasks. "[14]

Main components are concepts, strings, atoms, terms, and relations. Each of them can be identified by a unique identifier [14].

Concepts and Concepts Identifiers (CUI)

In the Metathesaurus concepts represent a meaning and meanings are never distinct. That is why Metathesaurus tries to merge all the synonyms for the same meaning together.

Each concept has a unique identifier (CUI). The CUI never changes, except when two or more CUIs refer to the same concept, in other words when new synonyms are found and one of the CUIs is obsolete.

Concept names and String Identifiers (SUI)

Each concept name and string has a unique identifier (SUI). For example, the same string in different languages will have different SUIs.

Atoms and Atoms Identifiers (AUI)

Every string and information from the source vocabulary is stored in atoms with a unique atoms identifier. For example if you have a string *Atrial Fibrillation*, then you have the atoms linking to the string from different source vocabularies.

Terms and Lexical Identifiers (LUI)

At the moment, terms and lexical identifiers are only available for the English language. Lexical variant or minor variations are stored together in one unit. This means that a LUI object can have attached several SUI objects.

Table 1 shows the usage of atoms, strings, terms, and concepts. *Atrial Fibrillation* appears in more than one source vocabulary and for every occurrence an AUI is given. They are both linked to a single SUI. The plural form of Atrial Fibrillation has a different string identifier, but since these are only lexical variations they are linked to the same term. There is also a different term (*Auricular Fibrillation*), but this term is seen as a synonym to *Atrial Fibrillations* and therefore it is linked to the same concept identifier (CUI).

Relationships and Relationship Identifier

There are many relationships between different concepts. They come from source vocabularies, Metathesaurus users, and NLM developers. There are two categories of relationships:

- **Intra-Source:** These relationships are implied by individual source vocabularies and represent the connection of context and concepts within the source. There are also statistical relationships computed by co-occurrence of records in the database. For example, they connect different concepts, like disease and drugs.
- **Inter-Source:** They represent the synonymous relationship in the Metathesaurus.

Every relationship has a unique relationship identifier (RUI).

Table 1 Hierarchical structure of UMLS[14]

Concept (CUI)	**Terms (LUIs)**	**Strings (SUIs)**	**Atoms (AUIs) * RRF Only**
C0004238 Atrial Fibrillation (preferred) Atrial Fibrillations Auricular Fibrillation Auricular Fibrillations	**L0004238** Atrial Fibrillation (preferred) Atrial Fibrillations	**S0016668** Atrial Fibrillation (preferred)	**A0027665** Atrial Fibrillation (from MSH) **A0027667** Atrial Fibrillation (from PSY)
		S0016669 Atrial Fibrillations	**A0027668** Atrial Fibrillations (from MSH)
	L0004327 (synonym) Auricular Fibrillation Auricular Fibrillations	**S0016899** Auricular Fibrillation (preferred)	**A0027930** Auricular Fibrillation (from PSY)
		S0016900 (plural variant) Auricular Fibrillations	**A0027932** Auricular Fibrillations (from MSH)

2.1.2 Semantic Network

The first idea of a semantic Network can be traced down to Aristotle according to Anderson and Bower [17]. It took quite some time until Semantic Networks were used for computers. In the early 1960s the first articles about Semantic Networks for computers were published.

In UMLS the main purpose of the Semantic Network is to provide additional information about the relationships of the concepts stored in the Metathesaurus and also to categorize all concepts. The Semantic Network contains 135 semantic types and 54 relationships. They are organized within a direct graph, where the semantic types represent the nodes and the relationships between them are the edges. Both the semantic types and the relationships have a hierarchical structure as shown in Figure 2-2 and Figure 2-3. Major groups of semantic types are organisms, anatomical structures, biologic functions, chemicals, events, physical objects, and concepts or ideas.

Each Metathesaurus concept is assigned at least one semantic type. The most specific semantic type in the Semantic Network is assigned to the concept. The accuracy of the assignment is varying. For example, a *chimpanzee* is categorized as *mammal* and not as a *primate*, because there is no specific type like primate.

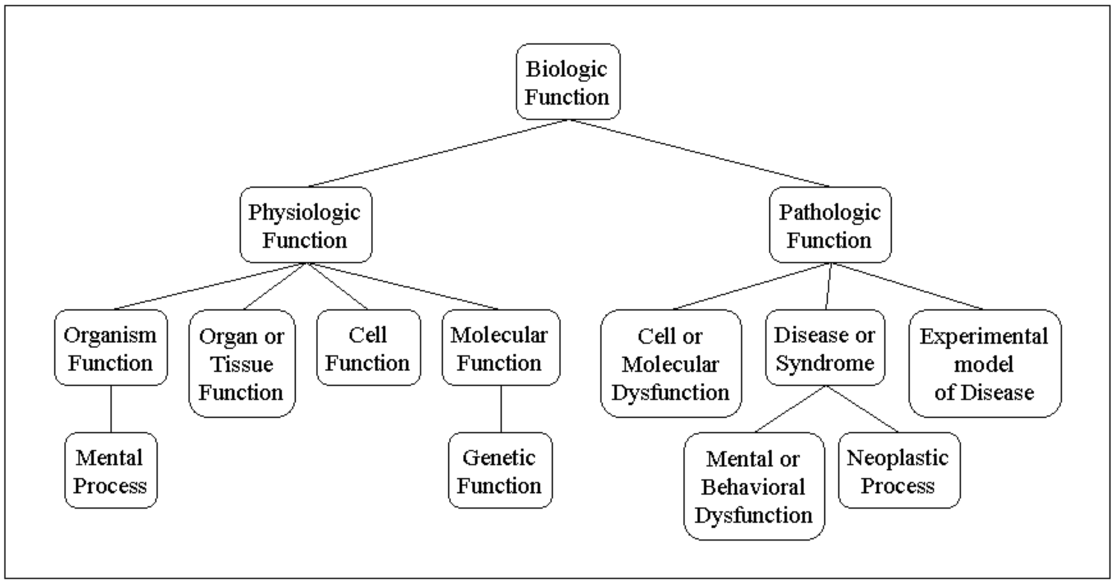


Figure 2-2 Hierarchical structure of semantic type “Biologic Function” [14].

Figure 2-2 is a small extract of the network displaying the semantic type *Biologic Function* with its children and grandchildren. Each child is connected by the “is-a” relation with its parent.

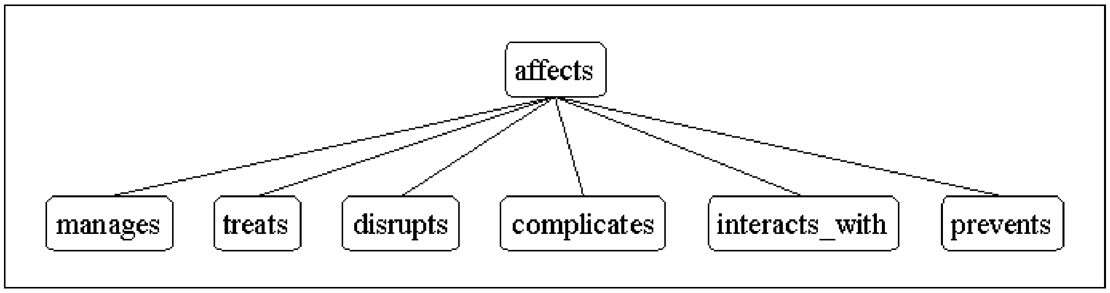


Figure 2-3 Hierarchical structure of relationship "affects" [14].

Figure 2-3 shows the relationship “affects” (functional relationship) and its children.

Nearly every high level semantic type is linked by a relationship as seen in Figure 2-4. Those relationships are between semantic types, and therefore not necessarily between all of the concepts and instances of those types. Furthermore, there are also cases where the inherited relationships are not logical and for this reason they are blocked. For example, the type ”mental process” cannot be linked to the type “plant” via the “process of” relationship, because plants are no sentient beings.

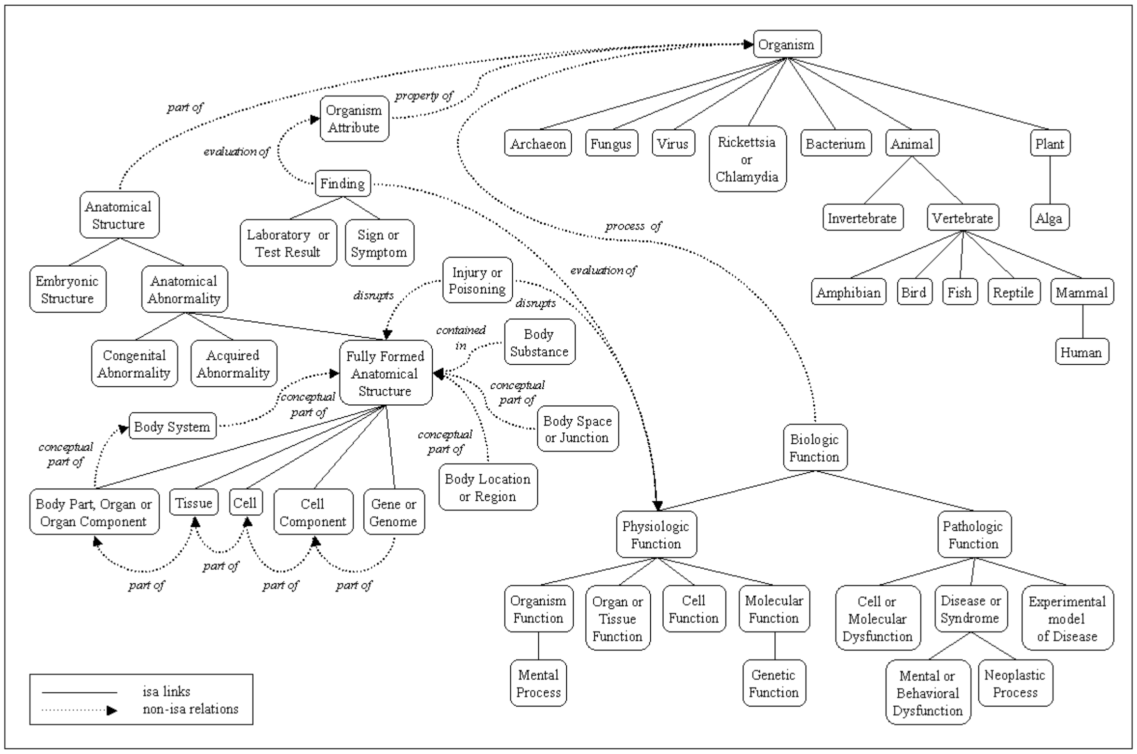


Figure 2-4 Part of the Semantic Network: a small section of the Semantic Network showing semantic types linked by relationships and the hierarchical structure of the semantic types [14].

2.1.3 SPECIALIST Lexicon

The lexicon includes biomedical and common English vocabularies. For every term in the lexicon the syntactic, morphological, and orthographic information is recorded. This information is necessary for the SPECIALIST Natural Language Processing (NLP) system. The lexical tools use the SPECIALIST NLP system to normalise strings, index words and find lexical variants [14].

The entries for the lexicon are obtained by different sources, for example Longman's Dictionary of Contemporary English, Dorland's Illustrated Medical Dictionary, Collins COBUILD Dictionary, The Oxford Advanced Learner's Dictionary, and Webster's Medical Desk Dictionary [14].

2.2 MetamorphoSYS

The Metathesaurus is an accumulation of a vast amount of data. Many information and concepts occur many times in this Metathesaurus, since they have been used in different source vocabularies. It is a fundamental characteristic of UMLS to keep all information of the source vocabularies. The sources are set together out of several hundreds of databases and languages. As one can recognize, it is not leading to success for any application to use the entire Metathesaurus, since it would take too much time for processing and searching through all the data. Furthermore, the received data would have to be processed once again, since it is possible to get pages of information for one term. Therefore, the NLM developed the MetamorphoSYS with the purpose to reduce the Metathesaurus on the user's requests.

2.2.1 Restriction Possibilities/Filter of the Metathesaurus

The MetamorphoSYS offers many possibilities how the amount of data can be reduced. The reduced database is then called Subset and is getting stored in the Rich-Release-Format (RRF), which has specifically been developed by NLM.

UMLS Licence Restriction: Unified Medical Language system (UMLS) has five grades of restrictions, because some sources are subject to costs and other restrictions.

Level 0: All Sources with level 0 can be used without license acquisition and are free of charge.

Level 1: It is not permitted to the user to translate parts of the sources into another language or to make derivatives.

Level 2: User is prohibited from using the vocabulary source in operational applications that create records or information containing data from the vocabulary source.

Level 3: Special licenses of the respective Sources must be purchased prior of their use. Research work may be executed in a restricted manner [1].

Level 4: UMLS sources with this level may only be used in the USA and are subject to special license conditions [2].

Language restrictions: With this option it is possible to select specific languages for specific sources (e.g., German, English, and French)

Input options: As a standard setting, the default values of the UMLS files are usually given, however alternatively it is also possible for an input to use predefined subsets and to limit and edit these once again.

Output options: There are two output possibilities, RRF and original release format (ORF). During the development of UMLS and the Metathesaurus, several rearrangements happened regarding the file format, so there is the possibility to output the files in the new or in the old format.

Another possibility to output the file is to create an additional Loadfile, which enables to load the data in a common database structure, for example MYSQL or Oracle.

Source list: Here, it is possible to select the source vocabularies for the subset. It has to be taken into account that only sources should be used, which one really needs. Otherwise, it comes to a blow up of the database with the negative effect of major unnecessary performance losses of the applications. Furthermore, sources, which contain misleading information, should get excluded. Source families should be carefully viewed as well, because the exclusion of a member may endanger the data integrity and finally contain information which have no meaning to the request.

Precedence: The meaning of the word already explains that with the help of this option, one can fix a preferential treatment. Depending which sources are most familiar to the user, those can be delivered as a standard concept. Example: We have two sources which identify the same concept but with a different naming. In actual fact we would like to return the information of the first source and ignore the information of the second source.

[1] For more detailed information see: http://www.nlm.nih.gov/research/umls/license.html#category3

[2] Further information can be obtained from http://www.nlm.nih.gov/research/umls/license.html#category4

Suppressibility: This is the counterpart to "Precedence". Here, one can suppress single sources or concepts, since this is in often easier than to prefer others.

Attributes filter: Single attributes can be removed from the subset of sources, since not all source attributes deliver usable information for the user.

Relationship filter: The different sources and concepts are connected with each other by relationships. One can specify which relationships one would not like to have in the subset and remove those. This can make the database structure easier and the access for the respective application.

Semantic type: For each concept one semantic type is assigned. In UMLS there are 135 semantic types altogether. As one can see in Figure 2-2, these are built up in hierarchical order. It is possible to remove certain types, which leads to the consequence that any child nodes are removed as well.

Example: If the semantic type *Molecular Function* will get removed, *Genetic Function* gets removed as well, this leads to the consequence that all concepts, which are assigned to these types, do not show up in the subset.

2.2.2 How to Use and View Subsets

The subset created by MetamorphoSYS will be stored in RRF. There are several possibilities to use and view these data:

1. To import the data with the help of the loadfile into a database
2. To view the data with the RRF browser
3. To search data with a text editor

These three possibilities offer different advantages and disadvantages, whereby the third option offers only one advantage: one can view it without great effort. Otherwise, this possibility has only disadvantages, since it is very difficult and confusing to edit and use these data. Therefore, I will not further refer to this possibility.

Import into the Database

With the help of the loadfile the RRF-Files can now be imported into a database. Depending on the size of the subset, it may take several minutes or even hours. After the successful data import, the database contains several spreadsheets and information. The next step is a bit more difficult to get closer to the data. Since there is no complete Entity Relationship (ER) diagram yet, the effort to analyze the individual spreadsheets and find out which information is useable is more complex. The tentative Entity diagram is surely helpful, as seen in Figure 2-5. The spreadsheets MRDOC and MRFILES contain additional information about the contents of the individual spreadsheets and attributes.

Advantages of this method:

- One can specify and fix the version and query very well.
- Queries about an interface are not a problem.

<u>Disadvantages of this method:</u>

- One must work out the database structure first.
- There are less viewing possibilities available, compared to the RRF browser.

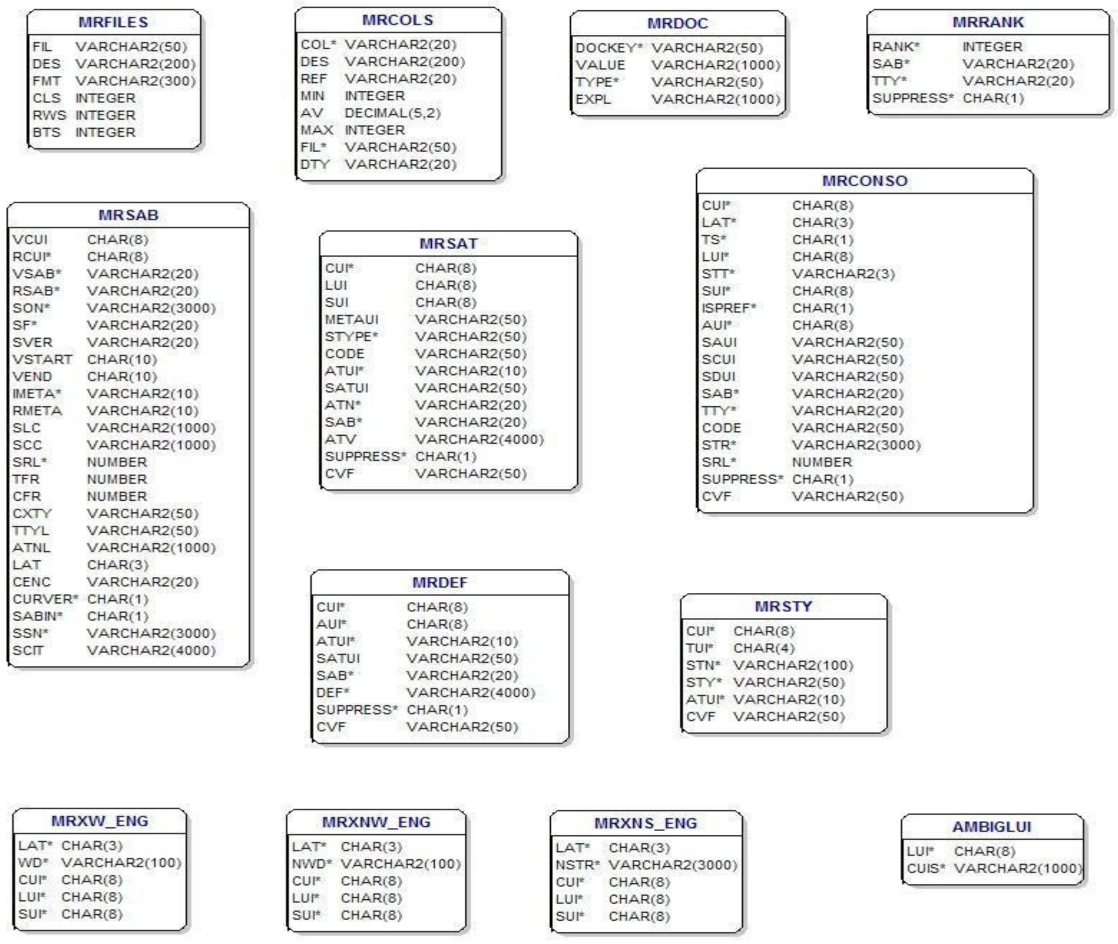
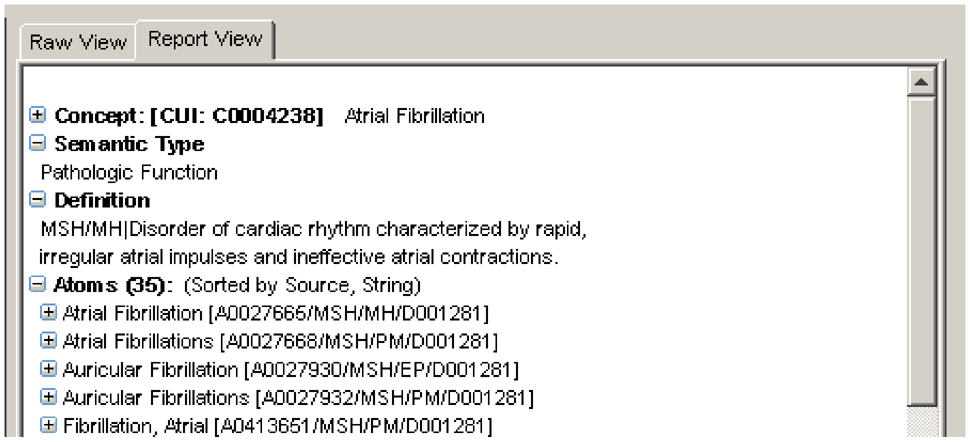


Figure 2-5 This is a small part of the tentative entity diagram of UMLS Metathesaurus [14].

RRF Browser

The RRF browser is a comfortable way to view the data of the subset. With the help of the RRF browser, the operator can look for any term or concept in the subset. The data provided in return are already prepared, as shown in Figure 2-6. On this example the search was made for *Atrial Fibrillation*. The result contains the semantic type as well as a definition of the term. In addition, all the atoms have been listed as well. These atoms contain the information in which source vocabularies the term appears, how they were written, their identification number, and furthermore additional internal data.



⊞ **Concept: [CUI: C0004238]** Atrial Fibrillation
⊟ **Semantic Type**
Pathologic Function
⊟ **Definition**
MSH/MH|Disorder of cardiac rhythm characterized by rapid, irregular atrial impulses and ineffective atrial contractions.
⊟ **Atoms (35):** (Sorted by Source, String)
⊞ Atrial Fibrillation [A0027665/MSH/MH/D001281]
⊞ Atrial Fibrillations [A0027668/MSH/PM/D001281]
⊞ Auricular Fibrillation [A0027930/MSH/EP/D001281]
⊞ Auricular Fibrillations [A0027932/MSH/PM/D001281]
⊞ Fibrillation, Atrial [A0413651/MSH/PM/D001281]

Figure 2-6 Result screen of the RRF-Browser after a successful search [14].

With the RRF browser it is possible to present the subset in a tree view. This makes it very easy to find the parent and child nodes.

However, a major disadvantage of the RRF browser is the lack of an application interface, which would enable the user to get closer to the desired data. Therefore, this method is rather useless for secondary applications.

2.3 Accessibility of UMLS

There are two possibilities how to access UMLS data. One option is to obtain the complete distribution stored on DVD and installing it on one's own network. The second possibility is to query the requested data decentralization, with the help of the UMLS Knowledge source server from the NLM Server.

2.3.1 Local Installation

Thereby, the UMLS data is installed locally on the licencee's computer. This makes it possible to create a subset with the help of the MetamorphoSYS and to use it for the respective application. The processing and access times are very short and reliable.

2.3.2 UMLS Knowledge Source Server

The UMLS Knowledge Source (UMLSKS) Server offers the possibility to query the data via the World Wide Web (WWW). The UMLSKS Server enables the access to all knowledge sources offered by UMLS, which are Metathesaurus, Semantic Network and Specialist Lexicon.

There are three possibilities to access the Knowledge Source Server.

1. Via a Web interface with the help of a Web browser
2. With the help of the Application Programmer Interface (API) which connects the user to the UMLSKS. This possibility is only implemented in Java
3. Via a TCP/IP socket interface, which permits access to the UMLSKS

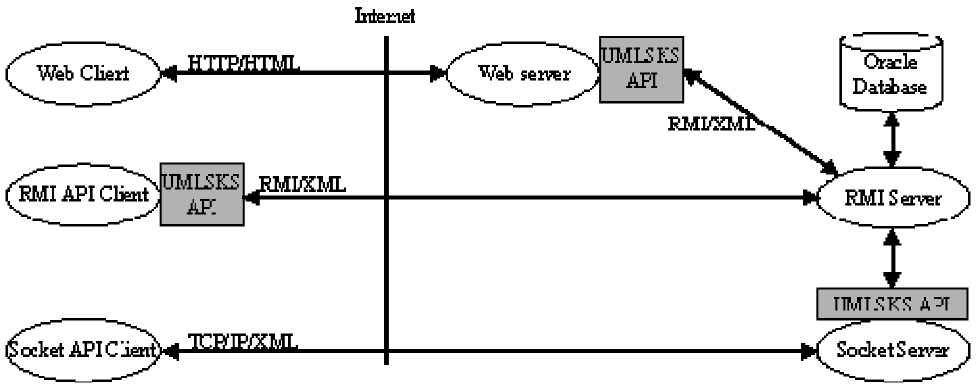


Figure 2-7 Diagram of the logical Architecture UMLSKS [14].

These different methods have of course different prerequisites.

Figure 2-7 shows that processing of the input data must be carried out first via a Web Server, prior of using the Web Interface for query. The most direct way is using the API functions, since these functions are placed directly without detours to the server and the data are delivered back.

With the TCP/IP's method the inputs have to be transmitted to the server as XML request and the server will return results in XML form.

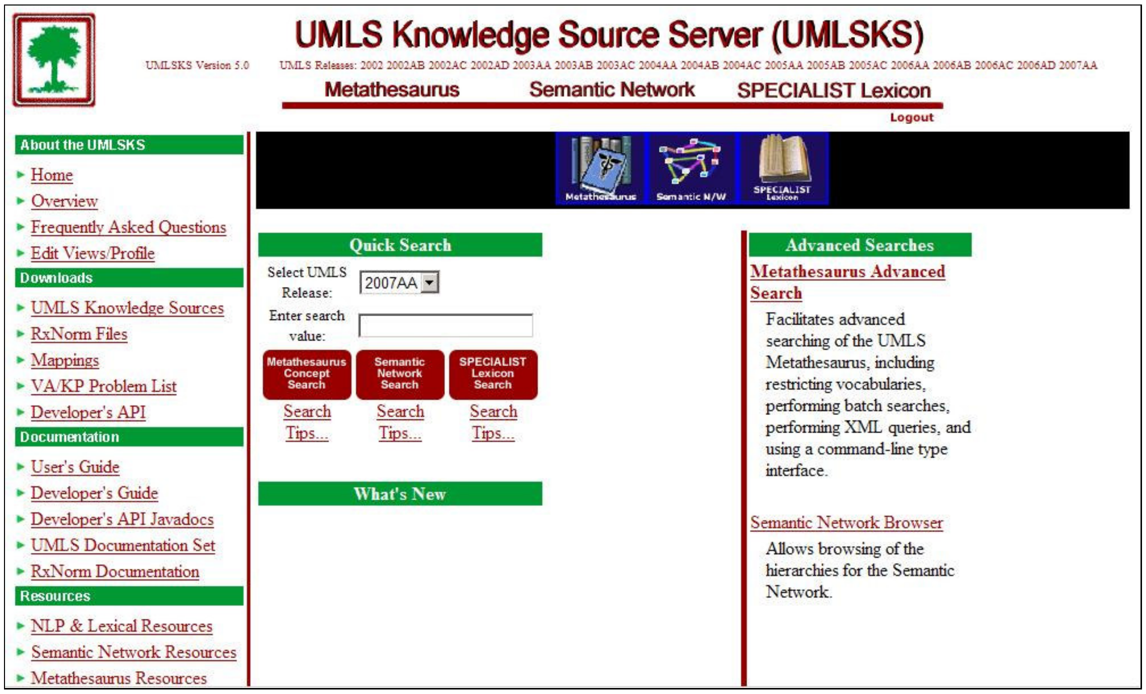
The data is changed at the Web Interface into HTML so that the Web browser can represent it. With the two other methods the data is returned in the Extensible Markup Language (XML).

Web-Interface

The Web interface offers a comfortable and simple access to the UMLS data. The access to the individual knowledge sources is very clear and efficiently presented (see Figure 2-8).

The queries for the individual terms and concepts are simply entered and evaluated in the provided query windows. The returned data is presented also in the browser, as seen in the example of the Specialist Lexicon in Figure 2-9.

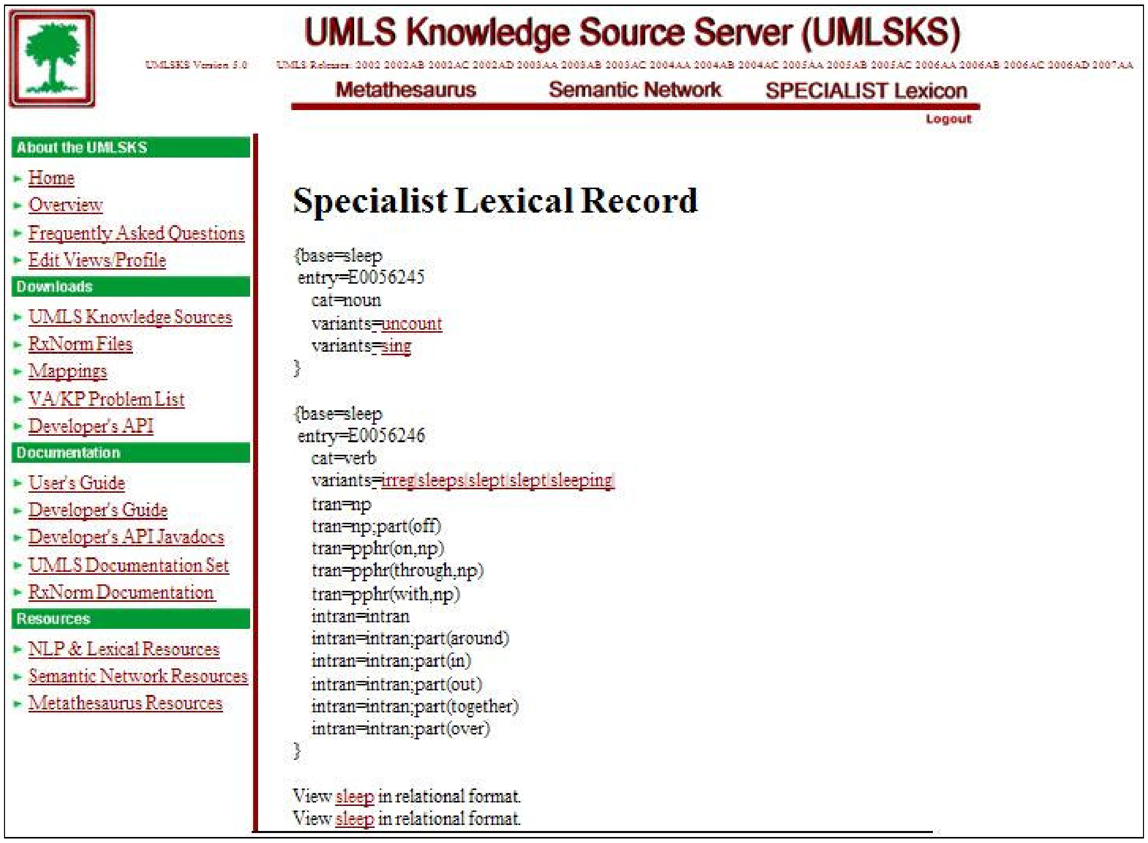
Besides the comfortable web interface, there is another possibility to send to the knowledge sources via a kind of command line for queries. The command lines Web interface is very well suitable for more specific queries and more details since one can precisely indicate the requests.



Facilitates advanced searching of the UMLS Metathesaurus, including restricting vocabularies, performing batch searches, performing XML queries, and using a command-line type interface.

Allows browsing of the hierarchies for the Semantic Network.

Figure 2-8 Screenshot of the Web interface with the different knowledge sources available.



Specialist Lexical Record

```
{base=sleep
entry=E0056245
   cat=noun
   variants=uncount
   variants=sing
}

{base=sleep
entry=E0056246
   cat=verb
   variants=irreg|sleeps|slept|slept|sleeping|
   tran=np
   tran=np;part(off)
   tran=pphr(on,np)
   tran=pphr(through,np)
   tran=pphr(with,np)
   intran=intran
   intran=intran;part(around)
   intran=intran;part(in)
   intran=intran;part(out)
   intran=intran;part(together)
   intran=intran;part(over)
}
```

View sleep in relational format.
View sleep in relational format.

Figure 2-9 Result Screen of a successful search with the Web Interface.

Application Programmer Interface with JAVA

UMLS provides some Java classes, which contain functions to make queries on the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon. The JAVA Remote Method Invocation (RMI) communications protocol is used. The RMI protocol works in such a way that makes it possible within a Java Virtual Machine for an object to call methods and functions into another Java Virtual Machine. Any arguments and parameters are handed over to the other virtual machine as well as the returned results and error. This enables the program to call functions and methods in the same way as if they would be stored on the local server. The KSSRetriever package is mainly used for this purpose and contains innumerable functions to fulfil the requests. Here are some examples:

- **GetCurrentUMLSVersion**: returns the version of the UMLS data:
- **FindCUI**: returns a list of Concept Unique Identfier (CUI), which comply with the input parameter.
- **GetSemanticType**: returns the Semantic Type of the searched Concept Unique Identfier (CUI).
- **ListSources**: returns a list of the Sources with a short description.

TCP/IP socket interface

After a connection has been made with a specific socket server over a specified port, an XML command to the UMLSKS is sent and XML data are received.

Generally, four different XML-Query types are distinguished.

1. **General UMLSKS details:** As the title is already indicating, version numbers and other items can be requested.

   Example: getCurrentUMLSVersion

```
<?xml version="1.0"?><getCurrentUMLSVersion version="1.0"/>
```

2. **Metathesaurus data:** requests to the Metathesaurus in XML form

   Example: findCUI

```
<?xml version="1.0"?><listDictionaries version="1.0"/>
```

3. **Semantic Network Data:** requests to the Semantic Network in XML form

   Example findSemType

```
<?xml version="1.0"?><findSemType version="1.0">
<release>RELEASE</release>      RELEASE means the optimal UMLS Release
<contains>STRING</contains>     STRING means the searched Semantic Type
<expandTree/>
</findSemType>
```

4. **Specialist Lexicon:** There is only one possible query: *getLecixalRecords*

```
<?xml version="1.0"?>
<getLexicalRecords version="1.0">
<release>RELEASE</release>    RELEASE means the optimal UMLS Release
<term>TERMNAME</term>         TERMNAME means the title/term searched for
</getLexicalRecords>
```

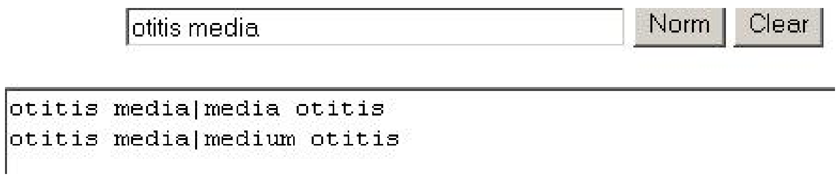
2.4 Specialist NLP Tools

The Specialist NLP Tools have been developed by The Lexical Systems Group of The Lister Hill National Centre for Biomedical Communications. The goal is to help application developers with lexical variation and text analysis. There are three tool packages, the Lexical Tools, Text Tools, and Spelling Tools.

2.4.1 Lexical Tools

These tools are a set of Java programmes specially designed to manage lexical variations. They use the Specialist Lexicon to fulfil their objectives. There are a total of five tools in this category [14]:

1. **Norm**: This tool creates a "normalized" output. This output is characterised by the lack of alphabetic case, inflection, spelling variants, punctuation, genitive markers, stop words, diacritics, ligatures, and word order. It also returns synonyms for the specified input and therefore it can return multiple outputs. There is a possibility to test the functionality of norm by a Web interface as seen in Figure 2-10.



```
otitis media|media otitis
otitis media|medium otitis
```

Figure 2-10 Example of functionality of NORM.

2. **LUINorm**: This tool nearly makes the same thing as Norm, except that it only returns a single uninflected output for any input. This process is called canonicalization. The side effect of this process is its greater inaccuracy than Norm, because of ignoring multiple forms.
3. **WordInd**: This tool breaks a string into a list of lowercased "words". Removing all punctuation and creating a word index.
4. **Lvg**: Lvg stands for Lexical Variant Generation. This tool generates, transforms, and filters lexical variants from the given input.
5. **Lgt**: Lgt stands for Lvg Gui Tools. This is a graphic user interface containing Norm, LuiNorm, WordInd, and Lvg. It is programmed in Java and runs on all Java platforms.

2.4.2 Text Tools

These tools are specially designed to analyze free text. They can tokenize strings into words, terms, phrases, sentences, and sections [14]:

There are a total of four tools.

1. **Tokenizer**: It can tokenize free text and Medline citation formats into sentences, words, and sections. A word is a sequence of alphanumeric characters, separated by white space or punctuation, which is also defined as a token. A sentence is found by looking at the punctuation and the capitalization of the following word. Sections are labelled document structures, for example title, abstract, introduction etc.
2. **LexicalLookup**: It is a term tokenizer, who retrieves terms from the Specialist Lexicon. For example, it would return "in vitro" as one token.
3. **Parser**: This parser is a minimal commitment barrier category parser and breaks sentences into phrases. For better visualisation follows an example sentence: "The study of phenolic compounds as natural antioxidants in wine…". This sentence is split into: "The Study", "of phenolic compounds", "as natural antioxidants", "in wine".
4. **VariantLookup**: This tool will give spelling variants, acronyms, synonyms, derivations, abbreviations and their expansions, and the inflections of each for a given term. For example, *sleep* would return: hypnic, sleep, sleeplessness, sleepy, sleeper, and more.

2.4.3 Spelling Tools

These tools are used to find close or related terms from an index.

- **GSpell**: Is a spelling suggestion tool to find close neighbours with various algorithms. It uses NGrams, metaphone, common misspellings, and homophone as retrieval tools.
- **BagOWords**: Is a phrase retrieval tool, which retrieves the closest matching phrase found in the Metathesaurus.
- **NGrams**: is a character based NGram retrieval tool that anchors the initial retrieval sets on the first and last NGrams of the query term. By default the NGrams are bi-grams.
- **MetaphoneRetrieve**: is a phonetic retrieval tool that uses Lawrence Philips Metaphone algorithm to normalize terms to a rough phonetic representation.
- **Homophones**: is a lookup mechanism into a table of homophones. Homophones are words, which are pronounced the same, but are written differently and have another meaning. E.g. accept, except
- **CommonMisspellings**: is a lookup tool to retrieve correct spellings of common misspellings that we have gleaned

3 Information Extraction

The development of IE as well as IR systems was substantially pushed forward by the Message Understanding Conferences (MUCs). The MUC-1 took place in 1987 and served mainly for the identification and the size of the information obtaining from documents. Since the format was not standardized for the storage of the attained results, it was not possible to compare the performances of the obtained systems with each other. This changed during the MUC-2 (1989), where templates were defined which had to be filled. Thus, it was possible to compare and to evaluate the individual results of the different systems with each other. Thereby, two terms were introduced, recall and precision.

The recall score stands for a measure of all correctly extracted information to all available information in the text.

The precision score stands for a measure of all correctly extracted information to all extracted information from the text.

Let's look at a sample text which contains five slots of information, which need to be extracted. The IE system returns two slots correctly; therefore, the recall score is 40%. The IE System extracts four slots in total, but only two of them are correct. Therefore, the precision score is 50%.

To get a rough feeling for the recall and precision score, I use the statement from Sundheim who states that “the human performance limits have not been scientifically determined, they are now estimated to be in the neighbourhood of 75% recall and 85% precision, assuming the All Templates scoring method and a representative test set.” [18] Furthermore, Sundheim shows that the leading system only fall 15% short by the recall measure and 30% by the precision measure.

Before the MUC-3 the recall and precision value were calculated as follows.

$$recall = \frac{Ncorrect}{Nkey}$$

$$precision = \frac{Ncorrect}{Ncorrect + Nincorrect}$$

Ncorrect … the correctly filled slots
Nincorrect … the incorrectly filled slots
Nkey … the complete number of available slots

Since the MUC-3 the calculation of the recall and precision values changed [19]. In Table 2 the scoring parameters are explained and the new formula is shown.

Table 2 Scoring Keys of MUC-3 [19]

COR	(CORRECT)	the number of correct slot fillers generated by the system
PAR	(PARTIAL)	the number of partially correct slot fillers generated by the system
INC	(INCORRECT)	the number of incorrect slot fillers generated by the system
SPU	(SPURIOUS)	the number of spurious slot fillers generated by the system
MIS	(MISSING)	the number of slot fillers erroneously not generated by the system

$$recall = \frac{Cor + 0.5 * Par}{Pos} \qquad precision = \frac{Cor + 0.5 * Par}{Act}$$

$$Pos = Cor + Inc + Par + Mis \qquad Act = Cor + Inc + Par + Spu$$

The choice of the texts to be analyzed varied from military messages in the area of fleet operations, up to newspaper analysis of terrorist activities. The texts to be analyzed on the last held MUC-7 where about plane crashes, spacecrafts and rocket launchings. Not only the texts are changing, but also the requirements become more and more complicated. The initial purpose was to fill out ten slots with information, which has been extended to up to 47 slots during the MUC-5. Not only the number of slots has changed, but also the structure of the templates has changed in the course of the time. With the new requirements of the MUC, also the system had to be changed, to be able to obtain the information.

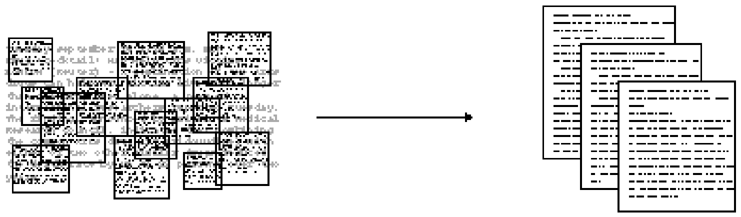


Figure 3-1 Schematic display of IR Systems [20].

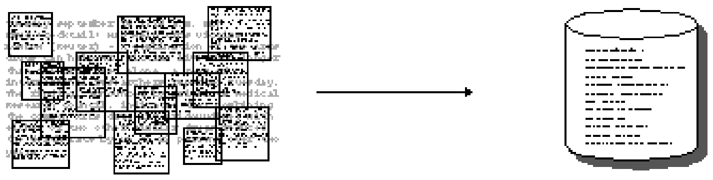


Figure 3-2 Schematic display of IE Systems [20].

IE is located in size and in the problem definition, between IR systems and NLP systems. "The purpose of an automatic retrieval strategy is to retrieve all the relevant documents at the same time retrieving as few of the non-relevant as possible." [21] As one can see in Figure 3-1 the main task of IR systems is the return of relevant documents. The functionality of IE Systems is shown in Figure 3-2. They search through documents as well, but out of those documents they extract specific information to populate a database or template. One can say, the texts found by the IR systems can be further analyzed in IE systems and enable a very powerful tool for text analysis [22]. A further difference also consists in the quantity and type of documents which are analyzed, whereby with IR systems the subject area of the documents is irrelevant. With IE systems, the documents must belong to the same subject area, otherwise no useable results can be provided. It is not an easy task to extract information from documents as easy as it may sound. A simple example demonstrates how the very same information can be represented in various ways [20]:

- BNC Holdings Inc named Ms G Torretta as the new chairman.
- Nicholas Andrews was succeeded by Gina Torretta as chairman of BNC Holdings Inc.
- Ms. Gina Torretta took the helm at BNC Holdings Inc.

- After a long boardroom struggle, Mr Andrews stepped down as chairman of BNC Holdings Inc. He was succeeded by Ms Torretta.

To recognize this information correctly, NLP is needed and used. NLP is a branch of artificial intelligence and deals with the analysis of the human language. On a simple example one can see how difficult it is to correctly interpret a simple three word sentence. The language used by humans is much more then just adding words and characters behind each other.

"Baby swallows Fly". To understand this sentence, it is insufficient just to know the words of this sentence. To understand it correctly, one needs more information; the context of the sentence is needed. The first meaning of this sentence is: that a human baby accidentally swallow a fly, on the other hand it could also mean that "baby swallows" have learned to fly [23, 24].

3.1 Two Approaches

There are two different ways how to design an IE System. Appelt refers to these two approaches as the knowledge engineering approach and the automatic learning approach [4].

3.1.1 Knowledge Engineering

In the centre of the knowledge engineering approach is the knowledge engineer. He must own special expertise about the IE system as well as knowledge how to create rules for the system. The creation and optimization is not a one time job, it requires many steps for construction and optimization of the rules and only numerous trials enable the knowledge engineer to refine the results. The knowledge engineer requires expert know how about the respective domain as well as a good intuition. Nothing else can be compensated for experience; this is one of the most important factors what a good knowledge engineer must bring along. Due to many times of rule adjustments the complete procedure is very time and work intensive.

3.1.2 Automatic Learning

Automatic Learning in comparison with knowledge engineering is something totally different. With automatic learning the system makes and modifies its own rules. It does not require somebody to write rules or to be an expert in IE systems. However, this person must be able to arrange a corpus of texts. For the IE system it is important to specify which information shall get extracted, so the IE system can compare the supplied results with the "optimum" results. As a result the system can adapt those rules.

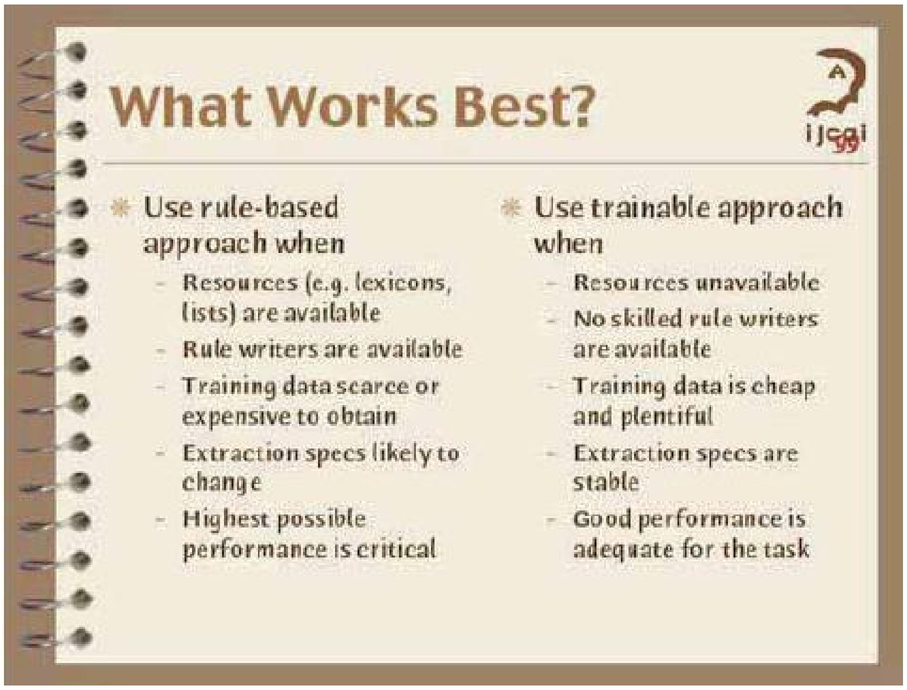
3.1.3 Choosing the Right Approach

Both approaches offer advantages and disadvantages. On the MUC, the IE systems which were produced with knowledge engineering achieved the best results up to now. Very effective rules can be created by a skilled, clever, and capable knowledge engineer, which can lead to very good results.

This advantage however comprehends already some disadvantages, because to create these rules it is very work intensive and time effortful. The demands on the knowledge engineer are also very intensive, since he must know a lot about both the domain and the IE system. Difficulties arise also, as soon as the specifications begin to change. This can lead to the circumstances that a large part of the set of rules must be remade, revised, or newly created.

The advantages and disadvantages for automatic learning are almost complementarily to those of knowledge engineering. At the automatic learning approach no specialist to analyze the documents is needed. It is sufficient to know the input and the desired output data. However, large quantities of training data are required to establish good rules automatically. It is not always possible to provide sufficient quantity of training data, or alternatively it is too expensive. A further disadvantage is that the system has to be completely retrained when the specifications changes.

Both procedural methods show disadvantages as soon as the specifications change. However, it depends substantially on the kind of change how much work or computation effort for the respective system is necessary.



What Works Best?

IJCAI 99

- Use rule-based approach when
  - Resources (e.g. lexicons, lists) are available
  - Rule writers are available
  - Training data scarce or expensive to obtain
  - Extraction specs likely to change
  - Highest possible performance is critical
- Use trainable approach when
  - Resources unavailable
  - No skilled rule writers are available
  - Training data is cheap and plentiful
  - Extraction specs are stable
  - Good performance is adequate for the task

Figure 3-3 Crucial circumstances to choose the correct approach [4].

Whatever approach is selected it significantly depends on the outer circumstances, which is shown in Figure 3-3.

3.2 Architecture

Typical IE systems consist of several components. Basic modules are represented in Figure 3-4 on the left column; these are *tokenization* of the input, *lexical and morphological processing*, a *syntactic analysis,* and a *domain-specific processing* [25]. The modules on the right side are optional elements, which are needed for some IE systems, whereas it depends on the respective application of the system. The optional modules are consisting of *text sectionizing and filtering*, *Part of Speech tagging*, *coreferencing*, and *merging partial results*. This represents only a choice of possible additional modules to the bare-bone of the IE systems; depending on the requirements other modules are needed [4, 25, 26].

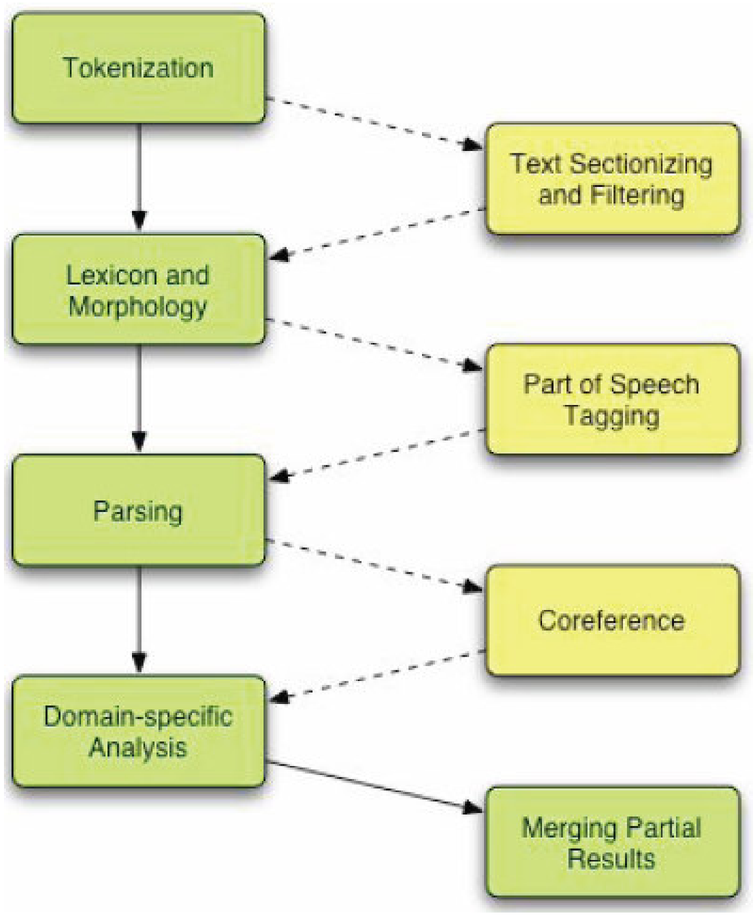


Figure 3-4 IE Architecture [25]

3.2.1 Tokenization

The main purpose of the tokenization module is to separate the documents in their individual parts so that these parts can be further processed. The break down can be carried out in sentences, words, or into single sentence sections. Terms, which contain several words as per example *Otitis media*, will get also separated here. With European languages this step is quite simple, since the words are separated by recognizable symbols such as blank, comma, hyphen etc. At languages such as Mandarin or Japanese the separation does not turn out so easily, since there is no blank or other clearly detachable symbol.

In Japanese “I am a student” means “私は学生である“. There are no spaces between “I” and “student”; “student” itself means “学生“. Another example to show the difficulties is: “My name is Michael” translated into Japanese this looks like this “私の名前はミハエルである“.

3.2.2 Text Sectionizing and Filtering

Text Sectionizing and Filtering is mainly used for languages where the individual words cannot be clearly separated, as it is the case with Mandarin or Japanese. Among other things this point also is called *Word Segmentation*.

3.2.3 Lexical and Morphological Analysis

The morphological analysis of texts depends on the language. Languages with simple inflectional morphology such as English do not require a morphological analysis. In this case a lexical analysis with a listing of all sorts of possible inflectional variants is sufficient here. In the German language it is more difficult because in German are a lot of compound nominal and derivation. Some examples: "Sprachwissenschaft" and "Linguistik" means the same in German, translated to English it means "linguistics". The first word is a compound of "Sprache (language)" and "Wissenschaft (science)". As a result of this most common dictionaries in German have 150,000 to 250,000 different entries and English dictionaries have 90,000 to 120,000.

As soon as the morphological analysis is completed, the lexical analysis can begin. The individual words are looked up in a lexicon to determine names, point of time, places, organizations and determining much more. The size of the lexicon is decisive, because larger does not mean automatically better. A too large lexicon can lead to an over-fitting and into a decrease of performance.

Domain specific lexicons can provide thoroughly better or equal good results with lower effort.

3.2.4 Part of Speech Tagging (POS)

POS is also called grammatical tagging and in English grammar are eight different parts of speech: noun, verb, adjective, adverb, pronoun, preposition, conjunction, and interjection. With this method words in a text get tagged to help with the reconnaissance of words with multiple meaning. Unknown words can get marked as well with the help of POS with certain likelihood. The usefulness of this tool is controversial; it provides advantages in many situations and it causes disadvantages in other cases, caused by wrong assignment of POS tags. No clear advantages could be recognized till now by using IE systems, which work with POS modules [26].

3.2.5 Syntactic Analysis / Parsing

This module analyzes the words and if necessary creates single term from several words, as one can see in Figure 3-5. The terms can be subdivided in three categories: Noun-group, Verb-group, and Particle. With the help of this subdivision it is possible to analyze relations between individual terms.

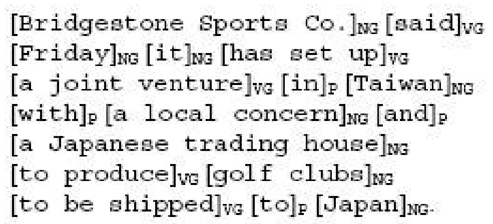

[Bridgestone Sports Co.]$_{NG}$ [said]$_{VG}$
[Friday]$_{NG}$ [it]$_{NG}$ [has set up]$_{VG}$
[a joint venture]$_{VG}$ [in]$_{P}$ [Taiwan]$_{NG}$
[with]$_{P}$ [a local concern]$_{NG}$ [and]$_{P}$
[a Japanese trading house]$_{NG}$
[to produce]$_{VG}$ [golf clubs]$_{NG}$
[to be shipped]$_{VG}$ [to]$_{P}$ [Japan]$_{NG}$.

Figure 3-5 Syntactic Analysis [26]

3.2.6 Coreferencing

During the MUC-6 a new task was introduced: the Coreference task. With the help of this module it is possible, as seen in Figure 3-5 to link pronouns with nouns. Example: *[Bridge-*

stone Sports Co.] with *[it].* Therefore, the relation gets allocated to the [it] and correctly referenced to the noun.

There are three basic coreference problems which should be solved with the help of this module [26].

1. **Name-alias coreference:** Synonyms of terms should get recognized and be handled as one term, e.g. “VW” and “Volkswagen”.
2. **Pronoun-antecedent coreference:** Pronouns should get recognized and equally linked with their antecedent, so the relations can be properly assigned, e.g. in Figure 3-5 [Bridgestone Sports Co.] with [it].
3. **Definite description coreference:** Different description of one term should get recognised and set in equal relation with them. An example for this would be: "Ford" and "the Detroit auto manufacturer". As beautifully this may sound it is in practice very hard to realise. To recognize and equate such terms, world knowledge would be required which, however, is very hard to implement.

3.2.7 Domain specific Analysis

The previous modules could for the most part be equally used for all IE-Systems. The IE-System is getting specifically trimmed for the required subject in this module. There are two approaches how this module can be realised.

- **Molecular approach:** It is primarily focused on generating high precision on cost of recall. This approach tries to fulfil its goal by matching most of the arguments to events in predefined patterns. It is also called “the standard knowledge engineering” approach.
- **Atomic approach:** It is primarily focused on generating high recall on cost of precision. “The basic idea is to assume that every noun phrase of the right sort and every verb of the right type, independently of the syntactic relations obtaining among them, indicates an event/relationship of interest.”[4]

There are different methods to represent the attained information. One of the simplest forms is to mark the found information in the text. Another representation form, which has been initially promoted by the MUC, is with the help of templates. Templates are attribute-value structures trying to represent information without the original text. The templates consist of a number of slots, in which the found data is stored.

3.2.8 Merging Partial Results

This module is only then necessary if the display format is more complex for the extracted information and/or is saved in several templates. There is no reason why templates should be complete which have been created by single phrases. Coherent information can be distributed over several sentences. Those multiple created templates are actually part of one single template. They must be identified and merged together. The identification of the templates is admittedly not trivial. Most IE systems, however, simplify this by joining templates together which show identical information in at least one slot.

4 JAVA Interface for UMLS

In my master thesis the main task is to make the UMLS ontology accessible for IE systems. UMLS itself provides a variety of tools, which are implemented in JAVA. Therefore, it was obvious to take JAVA as programming language to implement the code.

4.1 The main task

IE systems have multiple components. In general they are composed of five to eight parts (see Chapter 3.2). With UMLS it is possible to support some of those components, especially “Tokenization”, “Lexicon and Morphology”, and “Parsing”.

One requirement for the implementation was to be source independent. There is no restriction to any subject, such as radiology, genetics, therapy, or diagnosis. But even with a high limitation of the Metathesaurus, it is not possible to avoid multiple results for some phrases. This results in an ambiguous interpretation.

However, the application area of UMLS, as the name suggests, is limited to documents containing information of the medical area. But for medical documents it is a formidable source of information. With the help of UMLS and its tools sentences can be split into phrases, which can be searched within the UMLS Metathesaurus to obtain helpful information. Especially the classification of phrases with the help of the UMLS Semantic Network is very useful for the task "Domain-specific Analysis".

> *Example:* The term "patient" is equivalent to the concept of “Patients (CUI C0030705)” from MeSH vocabulary source. It has the semantic type "Patient or Disabled Group" assigned and from the semantic group "Living Beings".

The semantic type provides information, which can be decisive if the phrase is further processed or not from an interesting subject. UMLS provides even more information about the phrase and its semantic type. It is possible to search after the relationships of the specified phrase.

What is useful information within the Metathesaurus?

The Metathesaurus itself consists of 23 different entities if it is limited to the English language and all other languages are ignored. Each entity fulfils different goals und provides different information. But before it is possible to use the entities, it is necessary to fill them. This is a challenge on its own. As mentioned before UMLS consists of numerous source vocabularies (over 100). The problem is, if the sources are not limited to only what is needed, it would result in an extreme over fitting. Therefore, it is necessary to restrain the Metathesaurus with the help of MetaMorphoSYS to a subject.

UMLS provides different layers of information [27]. Figure 4-1 show a small extract of the Metathesaurus and Semantic Network layer. It shows how different concepts and semantic type are connected and related. A similar display method is used in the Semantic Navigator [27] to display the relationships of different concepts.

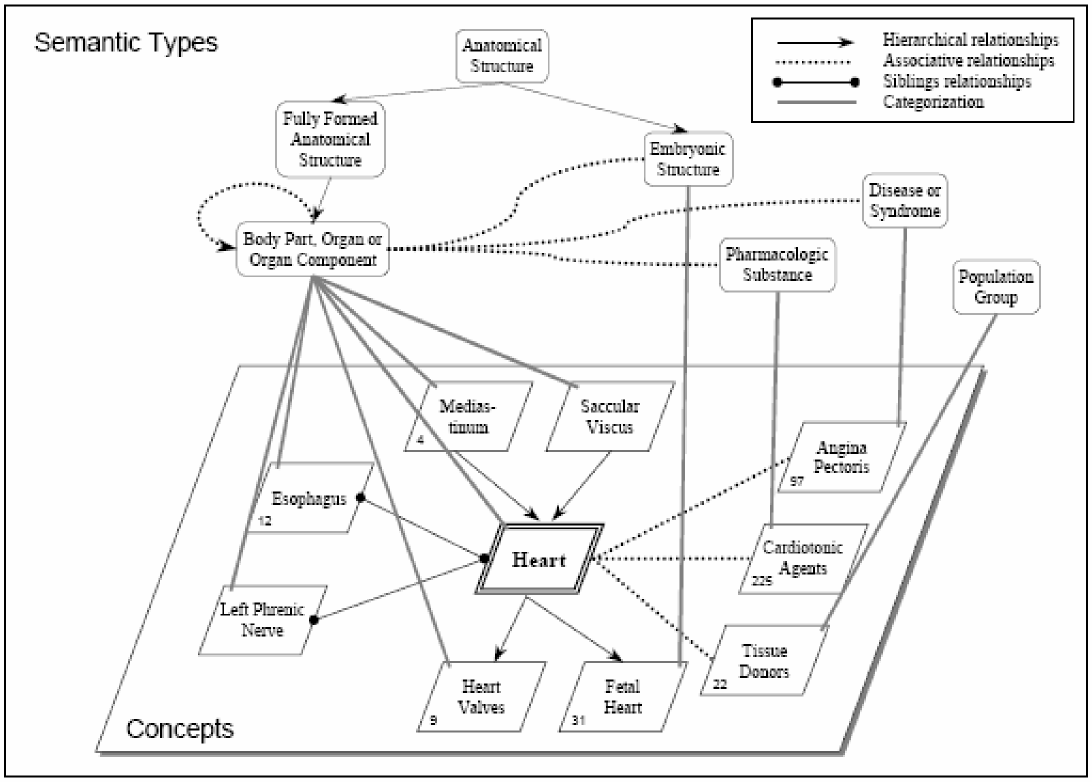


Figure 4-1 Schematic display of the UMLS layers [27]

For example, "otitis media" (C1) has a relation with "Middle ear infections and inflammations" (C2). The Metathesaurus returns C2 "parent of" C1 as connecting relation. But the Metathesaurus only has limited semantic capabilities which are highly dependent on the source vocabularies. Therefore, the Semantic Network returns for the same two phrases multiple relationships, such as "affected_by", "affects", "associated_with" etc. In Figure 4-2 the concept "Otitis media" is displayed with only its broader and narrower relations. A more detailed image of "Otitis media" is shown in Figure 4-3; this figure contains all relationships of this concept. Figure 4-4 shows the provided information of the Semantic Network about semantic type and its relations.

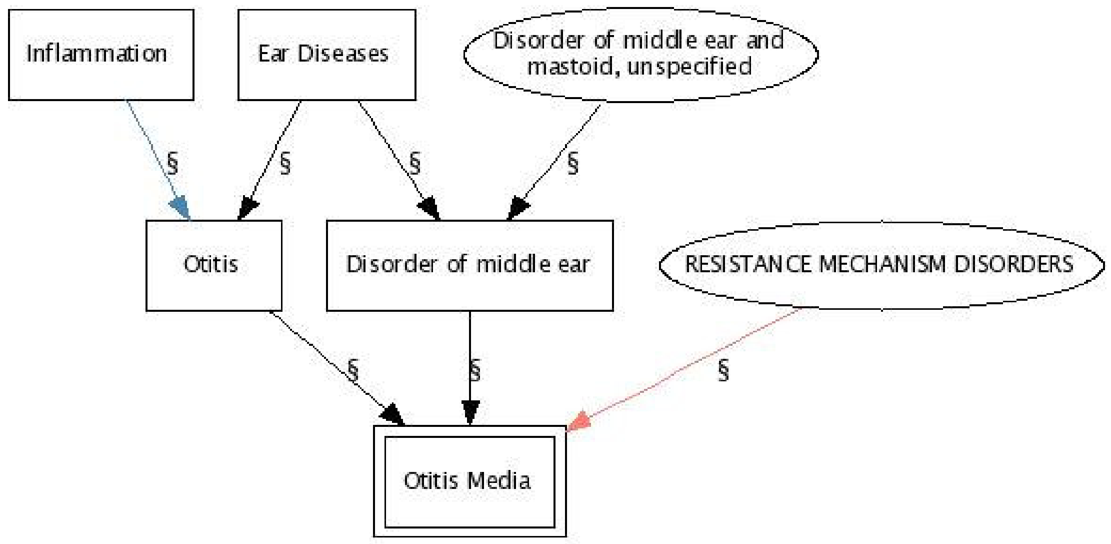

Inflammation
Ear Diseases
Disorder of middle ear and mastoid, unspecified
§
§
§
§
Otitis
Disorder of middle ear
RESISTANCE MECHANISM DISORDERS
§
§
§
Otitis Media

Figure 4-2 Broader and narrower relationships of “Otitis Media”

The square with the doubled frame represent the concept, after which is searched. The squares with single frame represent concepts found in the Metathesaurus and the ellipses stands for semantic types found in the Semantic Network

The coloured arrows represent the relations between the concepts and semantic types. The colours describe from which source the information about the relation is obtained.

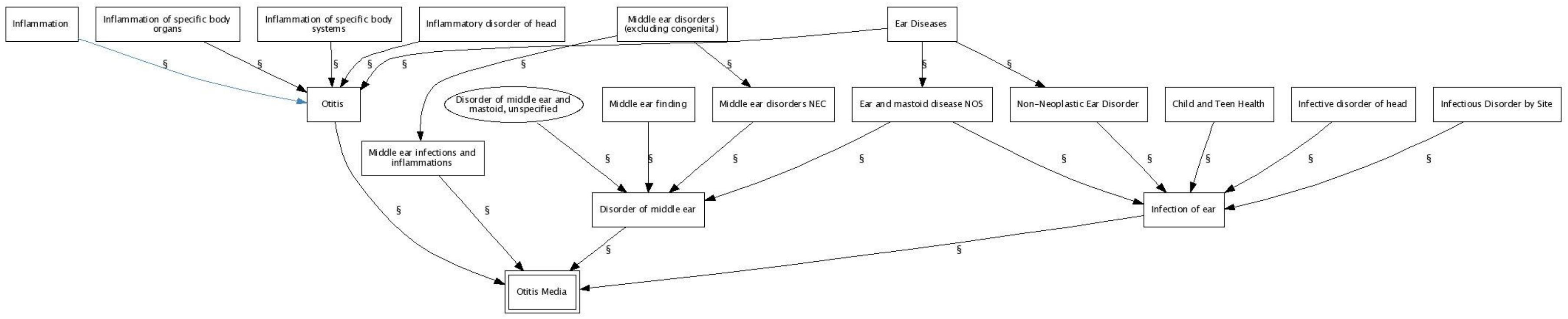

Inflammation
Inflammation of specific body organs
Inflammation of specific body systems
Inflammatory disorder of head
Middle ear disorders (excluding congenital)
Ear Diseases
Otitis
Disorder of middle ear and mastoid, unspecified
Middle ear finding
Middle ear disorders NEC
Ear and mastoid disease NOS
Non-Neoplastic Ear Disorder
Child and Teen Health
Infective disorder of head
Infectious Disorder by Site
Middle ear infections and inflammations
Disorder of middle ear
Infection of ear
Otitis Media
§

Figure 4-3 Complete map of relationships of "Otitis Media"

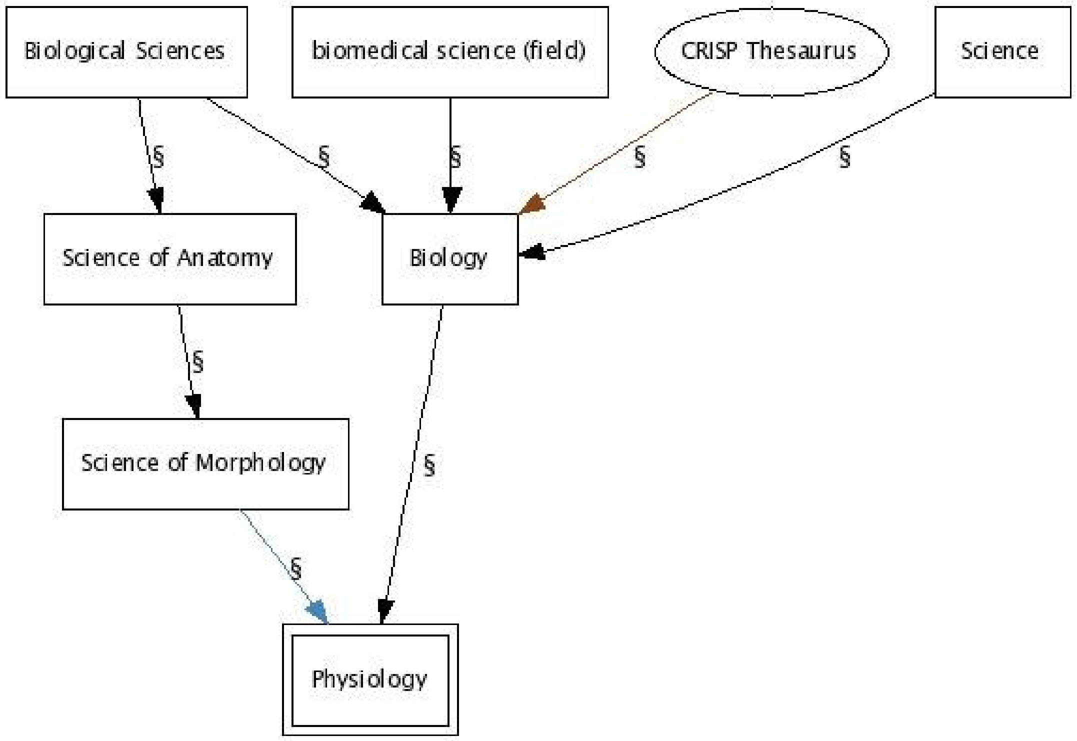


Figure 4-4 Relationships of the semantic type “Physiology”

There are lots of ways how to tokenize and parse sentences. To enable a high detection rate, whole sentences are used for input and UMLS tools are used to create phrases from the input.

I tried two different methods to compare the detection rate of the sentences. One method is by using the Lexical and Text Tools [28] of UMLS, the other one uses the MetaMap Transfer Tool (MMTx) [29]. With the created phrases the Metathesaurus is searched and all useful information is returned.

4.2 Tokenization / Parsing

In computer science, tokenization is a process to create tokens from an input string. Tokens are normally a block of text, which is useful for further processing. Tokenizing plain text means to split strings into words, separated by whitespaces. Whitespaces are "space", "tabulator", and "line-end".

Example: "*Information Extraction is an important task.*“
This string can be tokenized in "Information", "Extraction", "is", "an", "important", "task", “.”

In human languages, terms sometimes consist of more than one word. Therefore, it is important to create appropriate terms out of the tokens. This process is called parsing. It breaks sentences into phrases with the help of grammar.

With the obtained phrases it is now possible to search through the UMLS database and find additional information and represent them. In the following chapters the different implementations are presented.

4.2.1 Lexical and Text Tools

My first approach is with the help of Lexical and Text Tools provided by UMLS. The input sentence is tokenized with the help of Text Tools. Therefore, I used the *WordTokenizer* method. This method splits strings into word tokens. Individual pieces of punctuation are also considerer tokens as well. Multi word terms are also broken into multiple tokens; even real numbers get split up. Every string, which contains multiple tokens, is referred to as a sentence.

Table 3 Column definition for sentences

Sentence tag	Sentence Number	Begin Char offset	End Char Offset	Sentence String

Table 4 Column definition for phrases

Tag	Element Number	Begin Char offset	End Char offset	Phrase	Reduced Phrase	Number of Phrase Tokens	Has Head

Table 5 Column definition for tokens

Token tag	Token number	Begin Char offset	End Char Offset	Guessed Lexical Element Number (only has meaning for internal use)	Phrase token position (is -1 until sentence is parsed)	Token String	Token part of speech (is empty until Lexical Lookup or tagger client has been employed)

A few examples to explain how it works:

The string "1040-8398" is split into multiple tokens, with additional information.

Table 6 Example string "1040-8398"

Sentence	3	0	11	1040-8398		
Token	11	0	3	0	-1	1040
Token	12	4	4	0	-1	-
Token	13	5	8	1	-1	8398

After the tokenization the sentence is processed with the help of the *Parser* class. The parser provided by UMLS is a minimal commitment barrier category parser [30]. It is called TFA parser [31]. TFA stands for (T)homas Rindfleisch, (F)lorence Chang, and (A)llen Browne. The parser is a compilation of works from those three people, containing a minimal commitment parser (mincoMan), a barrier word parser, and some ideas from a barrier category parser.

Now, the tokens of a phrase can be combined to logical elements (phrases).

A free text sample:

"The study of phenolic compounds as natural antioxidants in wine

Plant phenolics present in fruit and vegetables"

This string was first tokenized into tokens containing only one word. They are now joined together into, for example, "phenolic compounds".

Table 7 Parser output for sample sentence

Phrase	0	4	12	The study	study	2	false
Phrase	1	14	34	of phenolic compounds	phenolic compounds	3	false
Phrase	2	36	58	as natural antioxidants	natural antioxidants	3	false
Phrase	3	60	67	in wine	wine	2	false
Phrase	5	0	22	Plant phenolics present	Plant phenolics present	3	false
Phrase	6	24	31	in fruit	fruit	2	false
Phrase	7	33	35	and	and	1	false
Phrase	8	37	47	vegetables	vegetables	1	false

The parser output adds additional information about the phrases, like phrase number, begin and end char offset, and the number of tokens.

To search the obtained phrases in the UMLS database it is necessary to normalize them first. With the help of the Lexical Tools especially a tool called Lexical Variant Generator (LVG) it is possible to normalize the phrases. The LVG tool has a lot of adjustable components for input, output, processing.

I use a parameter string like "-f:g:rs:o:t:q:q2:l:B:C:q4 -F:2 -R:1"

The first parameter "-f" stands for flow setup and defines what and how the input should be processed.

g	Remove genitive
rs	Remove plural patterns of (s), (es), and (ies)
o	Replace punctuation with spaces
t	Strip stop words
q	Strip diacritics
q2	Split ligatures
l	Lowercase
B	Uninflect words in a term
C	Canonicalize
q4	Get symbol names synonym

An output from LVG without the output parameters looks like

Table 8 Column definition for LVG output

Field 1	Field 2	Field 3	Field 4	Field 5	Field 6	Field 7+
Input	Output Term	Categories	Inflections	Flow History	Flow Number	Additional Information

Example output for the term "see".

Table 9 Sample output of LVG for "see"

see	Saw	2047	1	g+rs+o+t+q+q2+l+B+C+q4	1	

With the help of parameters "-F" and "-R"

-F:2	specifies which field should be returned
-R:1	restricts the number of variants returned

the output is trimmed down to only: "saw".

The obtained phrases are now ready to be searched after in the UMLS Database, which is more detailed explained in chapter 4.3.

4.2.2 MetaMap Transfer (MMTx)

The National Library of Medicine (NLM) initiated the project Semantic Knowledge Representation (SKR) [32] with the goal to manage information from free text. It is build up on the UMLS knowledge sources and its SPECIALIST tools to process natural language. A core developments is MetaMap [29].

The MMTx program is designed for biomedical researchers. It maps Metathesaurus concept to free text. It is composed of several modules. First the input text is parsed and tokenized. Lexical variants are generated from the resulting phrases. From the Metathesaurus, candidates are retrieved, who match the variants. The best candidates are evaluated by a final mapping algorithm and the candidate / or candidates with the highest ranking is/are returned. The goal is to represent the free text as good as possible with the found candidates.

The MetaMap Algorithm

It is composed of five modules: Parsing, Variant Generation, Candidate Retrieval, Candidate Evaluation, and Mapping Construction.

1. Parsing

   The parsing process is divided between two tools. Parsing is performed by the SPECIALIST minimal commitment parser [33], which processes the input string into noun phrases. Combined with the Xerox part-of-speech tagger [34], every phrase is a syntactic tag assigned, who has no unique tag from the SPECIALIST lexicon.

   The parsing process detects two noun phrases within the example text “ocular complications of myasthenia gravis”. The parser tries to find the most central part of the noun phrase and labels it as “head” of the phrase. *Complications* within “ocular complications" is the head of this phrase and *ocular* is a modifier. For later processing some words with specific tags are ignored, like prepositions, conjunctions, and determiners.

2. Variant Generation

   The phrase from the parser is subdivided into multiple words. Only single words or words that appear in the SPECIALIST Lexicon are processed, all others are ignored (like words that are prepositions, determiners, conjunctions, auxiliaries, modals, pronouns or punctuation). For example, the phrase "liquid crystal thermography" is split into "liquid crystal thermography", "liquid crystal", "liquid", "crystal", and "thermography" [35].

   Now for all substrings spellings, inflectional and derivational variants, acronyms and abbreviations, and synonyms are created after a procedure specified in Figure 4-5. Every variant gets a distance score from the generator and also a creation history.

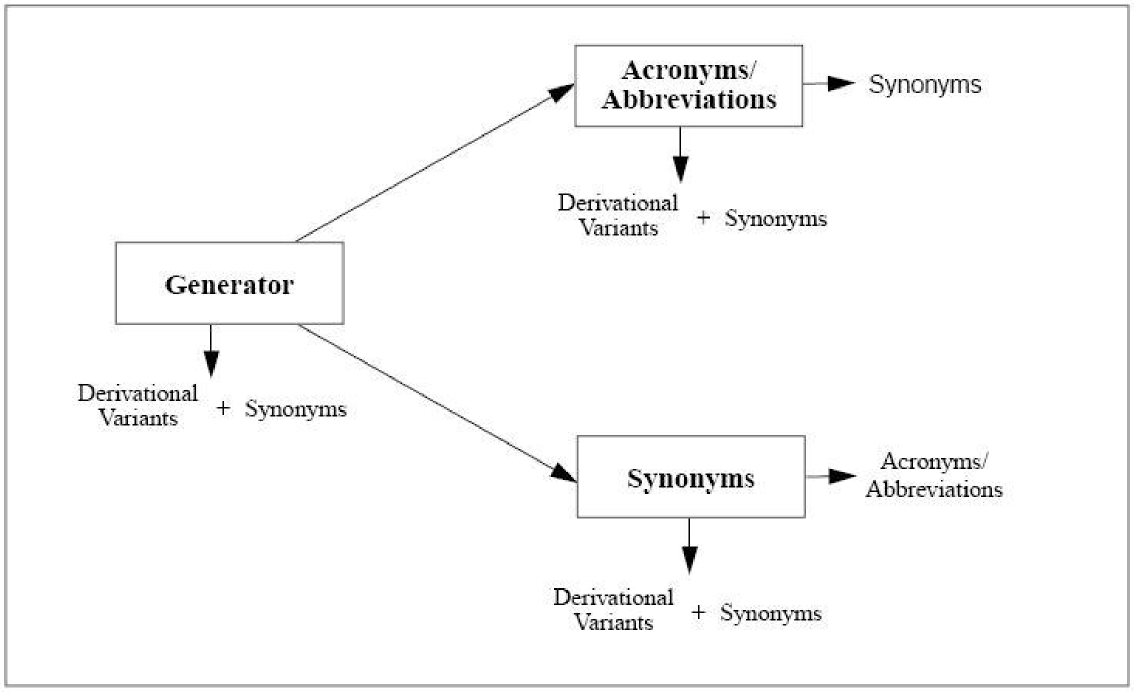
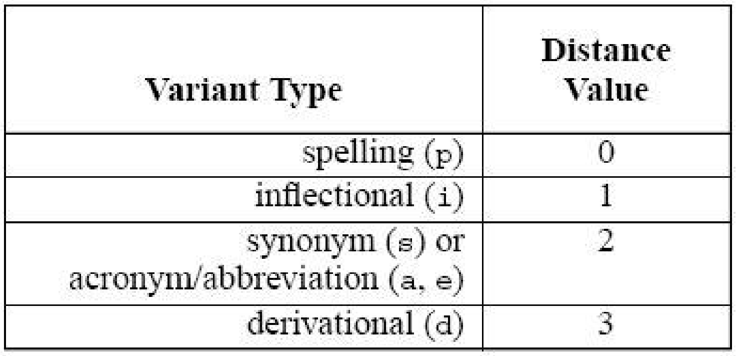

Acronyms/ Abbreviations
Synonyms
Derivational Variants + Synonyms
Generator
Derivational Variants + Synonyms
Synonyms
Acronyms/ Abbreviations
Derivational Variants + Synonyms

Figure 4-5 Variant generation [35]

   The creation history is composed of different labels. The label for a derivational variant is "d", the label for acronyms/abbreviations is "a", the label "i" is for inflectional, and the label for synonyms is "s". Now it is possible to mark every variant. How it is done in practice is shown in Figure 4-7. "Vision" is marked as "ssds" with a distance of nine, which means it is a synonym of "optical", "optical" is a derivation of "optic", "optic" is a synonym of "eye" and "eye" is a synonym of the source. In Figure 4-6 are the values for the distances. Our example "vision" is marked as "ssds", "s" has a value of two and "d" has a value of three. All labels summed together are the distance for this item, here it is nine.



Variant Type	Distance Value
spelling (`p`)	0
inflectional (`i`)	1
synonym (`s`) or acronym/abbreviation (`a`, `e`)	2
derivational (`d`)	3

Figure 4-6 Variant distances [35]

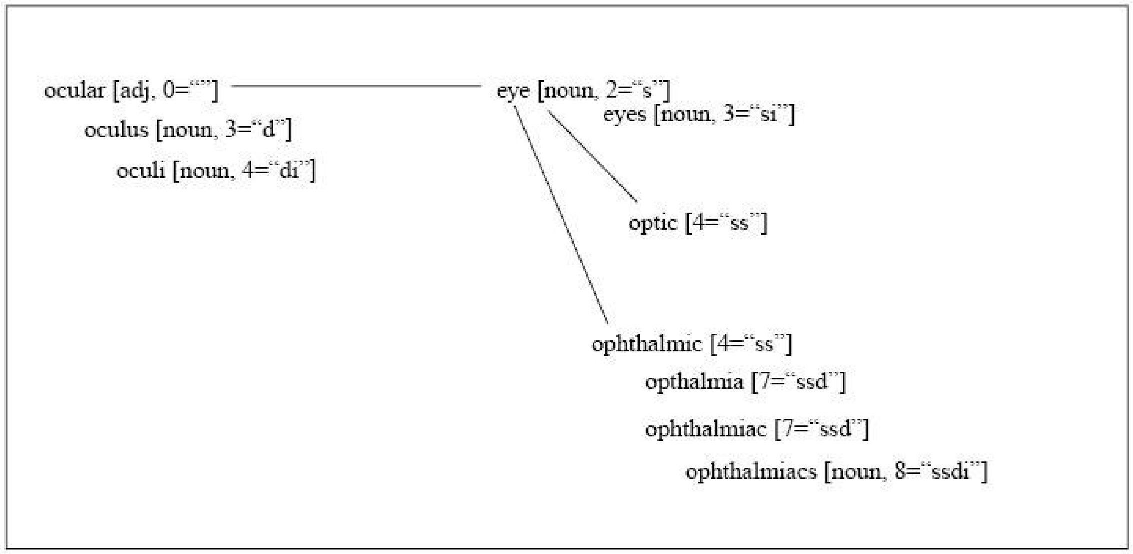


Figure 4-7 Variant generator for *ocular* [35]

Acronyms and abbreviations are not created with a recursive algorithm because it has shown that it produces incorrect results but derivational variants and synonyms are created recursively.

3. Candidate Retrieval

   This module works with the variants created prior in the module *Variant Generation*. Its primary goal is to find all Metathesaurus strings containing at least one phrase variant. For example, the phrase "of obstructive sleep apnea" returns for all variants approximately 130 Metathesaurus strings [36].

Variant string	Number of found candidates
"obstructive sleep apnea"	two candidates
"obstructive sleep apneas"	one candidate
"osa"	five candidates
"obstructive"	49 candidates

“sleep apnea”	one candidate
“sleep apnoea”	one candidate
“sleep”	58 candidates

And so on…

During normal MMTx process some candidates are filtered out before the evaluation. This applies to candidates who represent an overmatch or a concept gap. An overmatch is a candidate who has non-matching words on one end of the string. An example for the variant “sleep”, “sleep walking” is an overmatch and therefore ignored. A concept gap is similar to overmatch only that the non-matching words occur in the middle of the candidate. For example the candidate “‘Computerized Medical Record System” is a concept gap for “computer system”

4. Candidate Evaluation

This module calculates a measure for the match of the Metathesaurus candidate and the phrase of the input. Some candidates are already eliminated by the previous module by rejecting overmatch and concept gap. Now the remaining candidates are measured and evaluated by four components: centrality, variation, coverage, and cohesiveness. For each component is a normalized value between zero (weakest match) and one (strongest match) is computed. The final score for each candidate is calculated as following:

Mapping Score = (Centrality + Variation + 2*Coverage + 2*Cohesiveness)/6

It is then normalized to a value between 0 and 1000, where zero means no match and 1000 means perfect match. The weights were acquired by empirical results.

The **Centrality** value is one when the string contains part from the head of the phrase. Otherwise it is zero.

The **Variation (V)** value represents how much the phrase differs from the found candidate represented by a Metathesaurus string. V is computed after the formula:

$$V = \frac{\sum \frac{4}{D_i + 4}}{n}$$

D_i stands for the distance of word *I* and *n* for the number or words. Here are some examples to explain it in detail:

“oculi” has a distance D of 4. Which means after the formula V = 4/(4+4) = 0.5

“sleep apneas” has a distance D of [0,1]. V_1 = 4/(0+4) = 1, V_2 = 4/(1+4) = 0.8. V for the candidate is (1 + 0.8)/2 = 0.9

The **Coverage** value represent how good the phrase string matches the Metathesaurus string by comparing the numbers of involved words in the match. There are two important values, the *Metathesaurus span* and the *phrase span*. They are both computed very similar, the phrase span stands for the number of participating words from the phrase in the match, and the *Metathesaurus span* is the same with the Metathesaurus string. The coverage value now is the span divided by the total length, which means, *phrase span* divided by length of phrase and *Metathesaurus span* divided by the

Metathesaurus string length. The total Coverage is now computed an average between coverage of phrase and Metathesaurus with a weight of two for the Metathesaurus.

Some examples:

The original phrase is “obstructive sleep apnea”, its length is 3.

The candidate “obstructive sleep apnea” has a span of 3.

The matching Metathesaurus string also has both a span and length of 3.

The resulting coverage is $= (3/3 + 2*(3/3))/3 = 1$

For the candidate “sleep apnea” it looks different. Its coverage of the Metathesaurus string is 2/3, and the total coverage is $2/3 + 2*(2/2)/3 = 0.889$. The different Metathesaurus span results in the difference of the found candidate for “sleep apnea”.

The **Cohesiveness** value represents how many connected components are used in the match. A connected component is a maximal sequence of contiguous words participating in the match [37]. Similar to the coverage it is composed of two values. One value is computed for the phrase and one for the Metathesaurus string. The Cohesiveness is the square of the connected components divided by the square of the length of the string. Similar calculation for the Metathesaurus is made. The total Cohesiveness is the average of the two expressions with twice the weight for the Metathesaurus.

“Obstructive sleep apnea” has a value of $3^2/3^2$ and the Metathesaurus string has also $3^2/3^2$, therefore the total is $= (3^2/3^2 + 2*(3^2/3^2))/3 = 1$

“Sleep apnea” has a value of $2^2/3^2$ and the Metathesaurus sting has a value of $2^2/2^2$. The total is $= (2^2/3^2 + 2*(2^2/2^2)) = 0.815$

Now as mentioned above it is possible to calculate the score for each candidate with the formula (centrality + variation + 2*coverage + 2*cohesiveness)/6. With our examples for “sleep apnea” and “obstructive sleep apnea” a score of

$1000*(1.0 + 1.0 + 2*1.0 + 2*1.0)/6 = 1000$ for “obstructive sleep apnea” is reached. A score of $1000*(1.0 + 0.9 + 2*0.889 + 2*815)/6 = 884$ for “sleep apnea” is reached. For final mappings these scores differ slightly, because the cohesiveness value is calculated a little bit different.

If the option “ignore word order” is selected for the *Candidate Evaluation* the Coverage component changes to the *Involvement* component.

The **Involvement** value differs from the Coverage component by ignoring the word order, for example the Metathesaurus string “Lung Cancer” maps to the string “Advanced cancer of the lung”, which is composed of element advanced, cancer and lung, to “lung” and “cancer”. The coverage component would only match to “lung” and therefore creating a lower value. The final involvement value of the example is $(2/3 + 1)/2 = 0,833$.

5. Mapping Construction

   The input of this module is all evaluation candidates generated by the previous module. The main purpose of this final mapping is to find partial matchings and merge them together to the best possible match [38].

Table 10 Meta Candidates (8)

Score	Evaluation candidates	Semantic type
1000	Obstructive sleep apnoea (Sleep Apnea, Obstructive)	[Disease or Syndrome]
901	Apnea, Sleep (Sleep Apnea Syndromes)	[Disease or Syndrome]
827	Apnea	[Finding]
827	Obstructive (Obstructed)	[Functional Concept]
827	Sleep	[Functional Concept]
827	Sleep <3> (Sleep brand of diphenhydramine hydrochloride)	[Organic Chemical, Pharmacologic Substance]
755	Sleeplessness (Sleep Initiation and Maintenance Disorders)	[Mental or Behavioral Dysfunction, Sign or Symptom]
755	Sleepy	[Finding]

The algorithm is of a recursive nature and one of the most complex modules. All evaluation candidates are listed and sorted by their score. Now, the first candidate is taken, in our example "obstructive sleep apnea". It is already a complete match and therefore it is saved and we continue to the next entry. The algorithm searches through the rest of the candidates to look after another candidate to enhance the partial mapping. For example, for the partial mapping "sleep apnea" the only candidate, which has no overlapping with the mapping is "obstructive". The second complete mapping was found. Now, the algorithm cannot find any more candidates which it could add. It jumps to the next candidate and again tries to find candidates for its partial mapping. The algorithm is finished as soon as all evolution candidates are processed and only *complete mappings* are left.

The calculation of the score for the *complete mappings* only differs from the calculation of the candidate score by the cohesiveness component. As mentioned before the cohesiveness component uses connected components to compute the value. This is replaced with the length of the candidates. It offers a few advantages; overmatches and gaps are ignored for example.

Now the final mappings are returned with their score.

This algorithm returns best final mappings which are chosen by a score.

A nice example for a complete mapping with two partial mappings is "inferior vena caval stent filter". MMTx returns two concepts for this string "Vena Cava Filters" and "Stents". It has found both concepts even though they were mixed together.

4.2.3 Comparison

The first approach with the help of the Lexical and Text Tools is quite a simple one. It creates phrases and simplifies the expressions. Those expressions are looked after in the database with a simple search and the found information is returned.

But this approach is not always satisfying. In free text or, for example, in medical guidelines sometimes mixed or combined terms appear, which represent multiple concepts in the

Metathesaurus. Therefore, a simple match is not sufficient. The first approach is comparable with a “simple match”. There are different kinds of mappings [39]:

Simple match: This match occurs when the phrase exactly matches the Metathesaurus string.

Complex match: It appears when a phrase matches multiple Metathesaurus strings. For example the phrase “intensive care medicine”, matches to two Metathesaurus strings, “intensive care” and “medicine”.

Partial match: There are three different kinds of partial matches:

Normal Partial Match: Part of the phrase matches a Metathesaurus string without a gap. For example, the phrase “intensive care medicine” maps to the Metathesaurus string “intensive care”.

Gapped Partial Match: It means that parts of the phrase or Metathesaurus string are not involved in the mapping and the missing part is in the middle. For example “intensive medicine” is a gapped partial match for “intensive care medicine”.

Overmatch: An overmatch is like a gapped partial match, with the difference that the part not matching is at the beginning or the end of the Metathesaurus string. For example, the Metathesaurus string “intensive care medicine” is an overmatch for the phrase “intensive care”

No match: It means that no matching Metathesaurus string is found for the phrase or a part of it.

Table 11 Comparison of UMLS Tools and MMTx

	Implementation with UMLS Tools	**MMTx**
Simple Match	This approach can find all occurring simple matches.	This approach can find all occurring simple matches.
Complex Match	No complex matches are found, because the phrase is handled as one element	Complex matches are found, because the phrase is split up, into multiple variants. And each one is searched.
Partial Match	Only overmatch is possible, because a simple “occurs in” query is executed.	It is adjustable, what kind of partial matches are allowed. For example overmatch is generally turned off. All other partial matches are found

The first approach with only the UMLS tools is a very simple and quick solution. It is easily and quickly implemented. It is highly adjustable and extendable. The results of this method vary and are by far not as good as those of the MMTx. MMTx offer a nearly complete and configurable solution with a simple handling. The downside is that it is not easy to expand.

I tried both approaches and the findings of MMTx are more promising for my implementation. Therefore, I use the output of MMTx to conclude the implementation. With the output it is now possible to search the Metathesaurus and the Semantic Network, to provide and generate additional information.

4.3 UMLS

UMLS consists of three main knowledge sources, the Specialist Lexicon, the Metathesaurus, and the Semantic Network [16].

The parsed sentences from MMTx contain already information from the Specialist Lexicon. It also contains a CUI for every mapped phrase. With the help of this CUI, it is now possible to search in the Metathesaurus and the Semantic Network.

But before we are able to apply a search in the Metathesaurus, it is necessary to create a subset and arrange an environment with which we can easily access the information. As mentioned above the **Metathesaurus** is composed of 25 entities. Those entities are loaded into a database, and every entity is reflected as a table. Some important tables are:

- MRFILES: It contains information about the entities/tables
- MRCOLS: It contains information about the different abbreviations and attribute names
- MRDEF: It contains the definitions of all CUIs and in which source they are found.
- MRXNS_ENG: It contains for each CUI his full string. The information returned for a single CUI is most of the time ambiguous because of the nature of UMLS to preserve all information from the sources.
- MRCONSO: It contains information about the Concept, which Atoms are used, from which sources and so on.
- MRREL: It contains information about the connection or relationship between two Concepts.
- MRSTY: It contains the semantic type for the CUIs.

With this small selection of tables, basic information about the phrases can be found, but it has hardly any connection to the Semantic Network layer. Therefore it is necessary to create an additional database with the Semantic Network tables. The Semantic Network reduces the complexity of the Metathesaurus by assigning each concept at least one semantic type. They are organised hierarchically and linked together by relationships. But there are still 135 different semantic types. For some areas of applications it may be necessary to introduce smaller and narrower semantic grouping [40-42]. Therefore, the 135 semantic types are grouped into smaller semantic groups. In [41] 15 groups are formed after various aspects, such as semantic validity, parsimony, completeness, exclusivity, naturalness and utility. We used this approach in our UMLS implementation.

The **Semantic Network** consists of seven tables as shown in the tentative diagram in Figure 4-8:

1. SRDEF: It contains basic information about the semantic types and relations.
2. SRFIL: It contains descriptions of each table.
3. SRFLD: It contains detailed descriptions of all attributes used in the tables.
4. SRSTR: It contains the structure of the Network.
5. SRSTRE1: It contains the relationship of each semantic type, stored with unique identifiers.

6. SRSTRE2: It contains the relationship of each semantic type, stored with the full name.
7. SemGroups: It contains the 15 groups and which semantic type is referred to them.

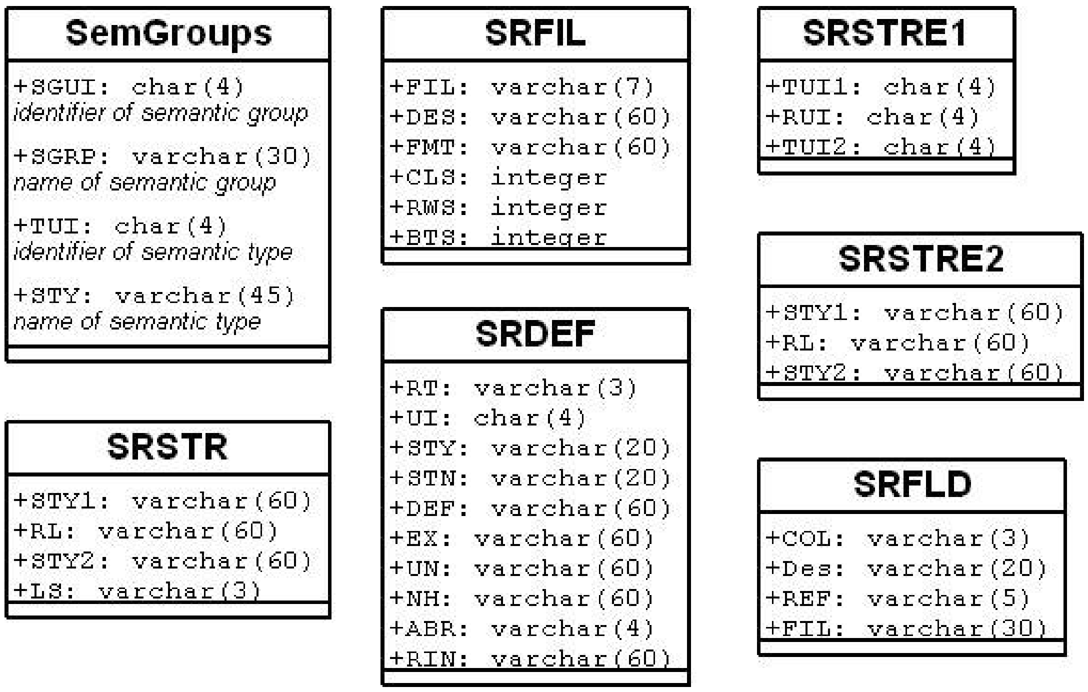


Figure 4-8 Tentative diagram of Semantic Network with Semantic Groups

With the information from the Metathesaurus, especially the semantic type, it is now possible to assign the correct Semantic Group and search after relations in the sentence.

For each processed sentence the number of phrases and the phrases are stored, also the position and relevance of them.

Each phrase in my implementation consists of numerous attributes, such as concept name, concept ID, semantic type. Detailed explanations of the phrase and sentence classes are in the next chapter.

4.4 Implementation

The main goal of the implementation is to create an interface for UMLS. The interface must be able to provide and pre-process the information gathered from the knowledge sources. It is realised in Java[3] and the package is named “UMLSint”, which is short for UMLS-Interface.

As mentioned above I tried two different approaches. The first approach is with the help of the Lexical Tools [16]. To realise the implementation different modules had to be installed and connected. It was necessary to install the Lexical and Text Tools from UMLS and include them into my code. Additionally a module to access MySql databases had to be installed.

The output fields from both implementations do not vary. The only things that diverse are the used tools and the parsing procedure. I only explain in detail the successful implementation

[3] http://java.sun.com/

with the MMTx tool. The MMTx tool already comes with the SPECIALIST Lexicon and slightly modified Lexical and Text Tools from UMLS.

My implementation consists of four classes:

1. **UMLSint:** This is the main class, which enables the connections to the databases and retrieves all information.
2. **UMLSSentence:** This class is created to mange and process the input string on the sentence level.
3. **UMLSPhrase:** This class stores the extracted phrase and handles all candidates from the database regarding its meaning.
4. **UMLSCandidate:** This class stores information about the candidates, such as concept id, term type, semantic type, and so on.

Additionally, we defined another class:

- **UMLSintDemo:** This is a demonstration code class, which shows the basic functions of UMLSint and displays the gathered information. The demo class has two different execution ways, one is processing a string and the other one is processing a file.

The configuration settings to locate and enable the access to the UMLS resources (i.e., Metathesaurus and the Semantic Network) can be accomplished by a property file.

As mentioned before, the main task is to gather and retrieve information for phrases. An additional functionality is retrieving information regarding relationships between phrases and concepts as well. Also a function for "blank" queries to the database is realised, which means it is possible to execute SQL statements within the code.

The UMLSint class handles all requests as seen in Figure 4-14. UMLSint receives a sentence and creates an UMLSSentence object from it. The input is parsed and analysed with the help of the MMTx tool. With the obtained phrases UMLSPhrase objects are created and associated with the corresponding sentence. Then the additional information from the database is loaded into the UMLSCandidate objects.

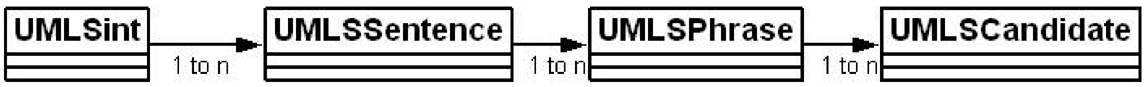


Figure 4-9 Class diagram of the UMLSint package

4.4.1 UMLSint

The **UMLSint** class enables the connection to the databases and fills the other objects with information. In Figure 4-10 the various methods and parameters are displayed. The parameters are obtained by the property file. It holds information about the database and the connection details.

There are several methods which enable the class to operate:

- UMLSSentence **setUMLSSentence** (String sentence)
  It returns an *UMLSSentence* object, which contains the found information for this sentence and its containing phrases.

- Vector **umlsParse** (String sentence)
  The string object contains a sentence or a sequence of words. The method analyses the input and detects phrases by utilizing MetaMap Transfer (MMTx). It returns a vector of found phrases as *UMLSPhrase* objects.

- UMLSPhrase **fillPhrase** (UMLSPhrase up)
  The method retrieves UMLS information for the UMLSPhrase object, adds the information to the object and returns it.

- Vector **findRel** (String cui)
  The parameter "cui" is a string containing the CUI (i.e., concept unique identifier), which is the primary identification key for every concept. The method returns a vector of strings, which contains the relations possible for this concept. The string is composed of several attributes: concept id of a connected concept (CUI2), name of the concept (NSTR), name of the relationship (REL), relationship attributes (RELA), relationship id (RUI), source id (SAB), source of the relationship label (SL), and relationship group (RG).

- Vector **getRel** (String cui1, String cui2)
  The parameters "cui1" and "cui2" are both strings, each containing a CUI. The method now searches through the Metathesaurus database to find all relations between the two specified concepts. It returns a vector of strings containing the relationships.

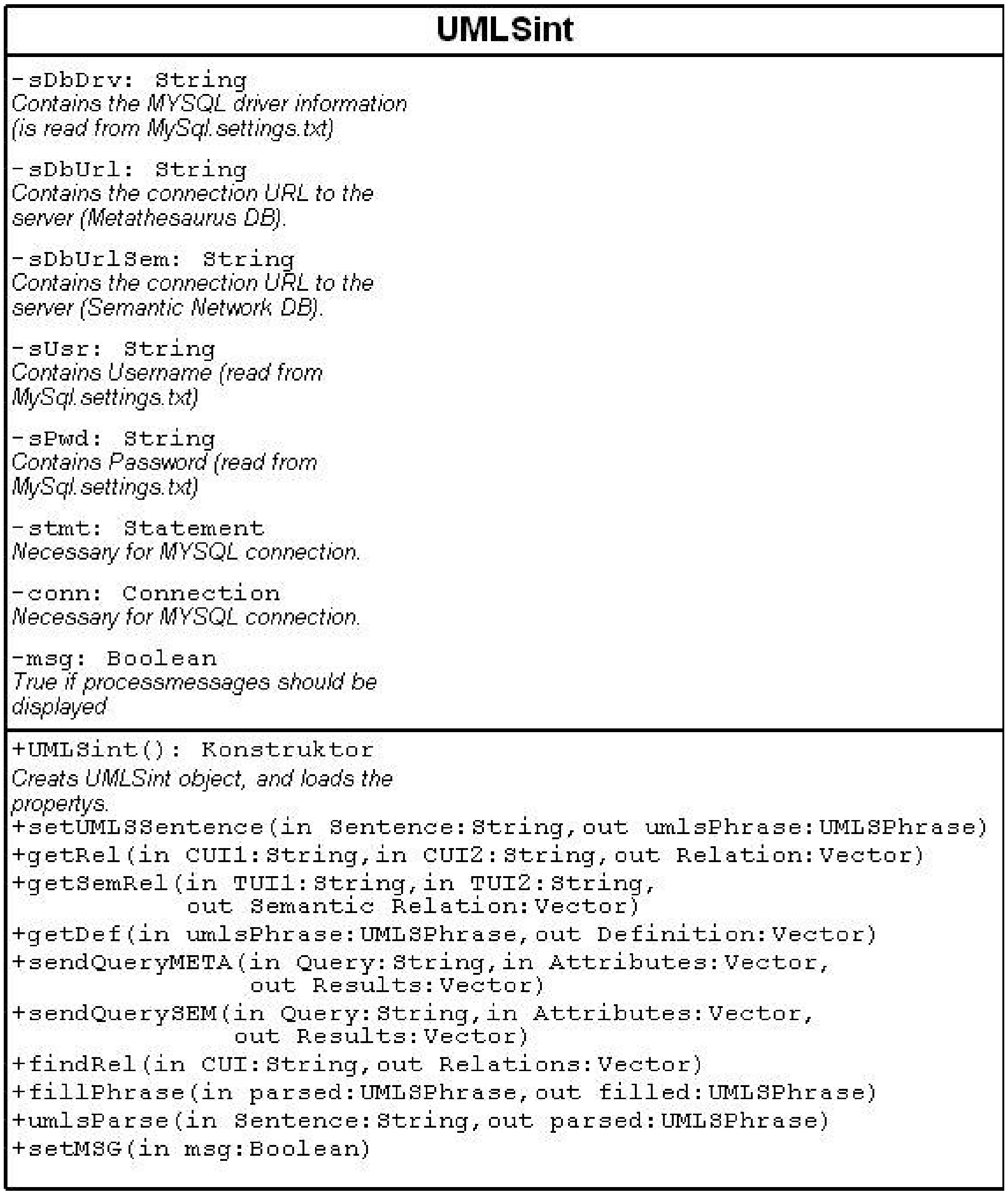
- Vector **getSemRel** (String cui1, String cui2)
  This method searches for semantic relationships between two concepts.

- Vector **getDef** (UMLSPhrase up)
  This method returns for all candidates of this phrase their definitions.

- Vector **sendQueryMeta** (String query, Vector att)
  The parameter "query" is a string containing an SQL query. The second parameter "att" is a vector of strings, where each string represents an attribute in the Metathesaurus database tables. The method returns the query's results.

- Vector **sendQuerySem** (String query, Vector att)
  This method is sends a query to the Semantic Network database. The parameter "att" contains strings representing attributes in the Semantic Network database tables.

- void **setMsg** (Boolean set)
  The parameter "set" is a boolean value, which specifies whether processing messages should be displayed or not.



```
UMLSint
-sDbDrv: String
Contains the MYSQL driver information
(is read from MySql.settings.txt)
-sDbUrl: String
Contains the connection URL to the
server (Metathesaurus DB).
-sDbUrlSem: String
Contains the connection URL to the
server (Semantic Network DB).
-sUsr: String
Contains Username (read from
MySql.settings.txt)
-sPwd: String
Contains Password (read from
MySql.settings.txt)
-stmt: Statement
Necessary for MYSQL connection.
-conn: Connection
Necessary for MYSQL connection.
-msg: Boolean
True if processmessages should be
displayed
+UMLSint(): Konstruktor
Creats UMLSint object, and loads the
propertys.
+setUMLSSentence(in Sentence:String,out umlsPhrase:UMLSPhrase)
+getRel(in CUI1:String,in CUI2:String,out Relation:Vector)
+getSemRel(in TUI1:String,in TUI2:String,
           out Semantic Relation:Vector)
+getDef(in umlsPhrase:UMLSPhrase,out Definition:Vector)
+sendQueryMETA(in Query:String,in Attributes:Vector,
               out Results:Vector)
+sendQuerySEM(in Query:String,in Attributes:Vector,
              out Results:Vector)
+findRel(in CUI:String,out Relations:Vector)
+fillPhrase(in parsed:UMLSPhrase,out filled:UMLSPhrase)
+umlsParse(in Sentence:String,out parsed:UMLSPhrase)
+setMSG(in msg:Boolean)
```

Figure 4-10 Class diagram of UMLSint

4.4.2 UMLSSentence

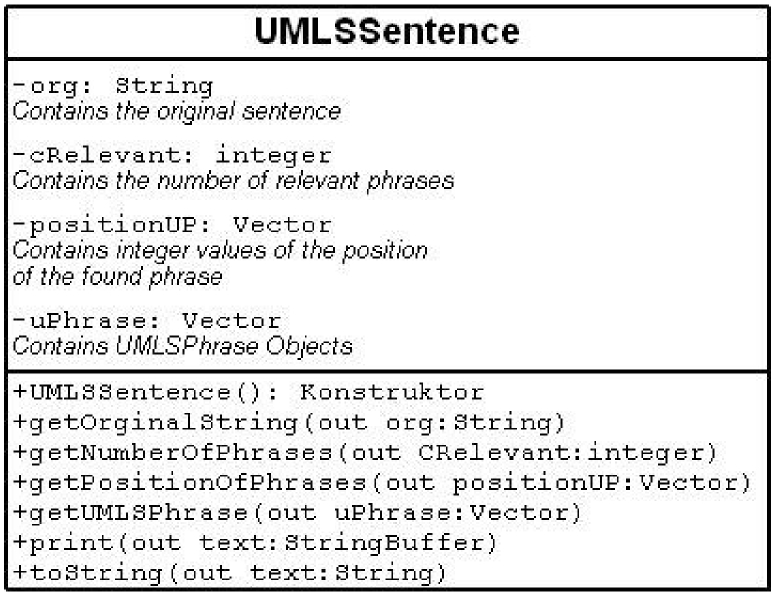
The **UMLSSentence** class stores the sentences and the parsed UMLSPhrase objects with the UMLS information. It contains four methods as seen in Figure 4-11:

- String **getOrginalString** ()
  It returns the original sentence as a string.

- Integer **getNumberOfPhrases** ()
  It returns the number of phrases found in the sentence

- Vector **getPositionOfPhrases** ()
  It returns a Vector, which contains the position of each phrase in the sentence.

- Vector **getUMLSPhrase** ()
  It returns a Vector with the *UMLSPhrase* objects.



```
UMLSSentence
-org: String
Contains the original sentence
-cRelevant: integer
Contains the number of relevant phrases
-positionUP: Vector
Contains integer values of the position
of the found phrase
-uPhrase: Vector
Contains UMLSPhrase Objects
+UMLSSentence(): Konstruktor
+getOrginalString(out org:String)
+getNumberOfPhrases(out CRelevant:integer)
+getPositionOfPhrases(out positionUP:Vector)
+getUMLSPhrase(out uPhrase:Vector)
+print(out text:StringBuffer)
+toString(out text:String)
```

Figure 4-11 Class diagram of UMLSSentence

4.4.3 UMLSPhrase

The **UMLSPhrase** class is an important information storage object. It consists of seven data elements, which can be seen in Figure 4-12. In the same figure is also a schematic display on how the *UMLSCandidates* are stored in the *UMLSPhrase*. Due the nested nature of the UMLS multidimensional vectors are necessary.

Every phrase can be mapped to multiple concepts. And also the concepts can be a combination of concepts. For each concept an *UMLSCandidate* object is created. The *UMLSCandidate* stores the semantic type, source type, concept ID.

Description of the methods and their fields:

- String **getOrginalString** ()
  This method returns the original string from the phrase.

- String **getPosition** ()
  This method returns the position of the phrase in the sentence.

- Boolean **getRelevant** ()
  It returns true if the phrase found has additional information in the UMLS (depending on the sources used).

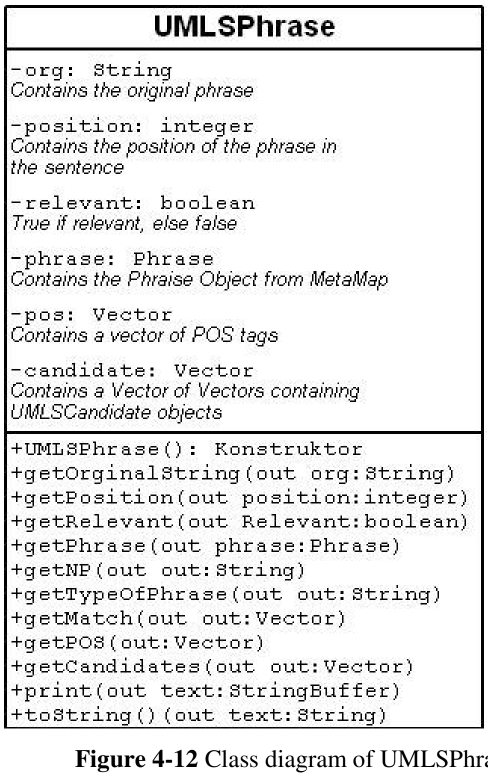
- Phrase **getPhrase** ()
  This method returns the MMTx Phrase object created during the parsing procedure.

- String **getNP** ()
  It returns the noun phrase of the original phrase. For example, the original phrase is "with fewer side-effects", the corresponding noun phrase is "fewer side-effects".

- String **getTypeOfPhrase** ()
  It returns the classification of the phrase. For example: "prep_phrase" stands for preposition phrase.

- Vector **getPOS** ()
  This method returns a vector with the Part Of Speech tags for each token in the phrase.

- Vector **getCandidates**()
  This method returns a vector of a vector containing UMLSCandidate objects, as shown in Figure 4-12. This is necessary, because of the ambiguity of UMLS. For example, the phrase "individual patients" is a complex match. Therefore, there are two matching concepts, one for "individual" and one for "patients". Furthermore there are more different complex matches, because there are many matching concepts for "individual" and "patients" and all combinations are valid. The first vector holds the complex matches and the second vector holds the possible combination of the candidates.



```
UMLSPhrase
-org: String
Contains the original phrase
-position: integer
Contains the position of the phrase in
the sentence
-relevant: boolean
True if relevant, else false
-phrase: Phrase
Contains the Phraise Object from MetaMap
-pos: Vector
Contains a vector of POS tags
-candidate: Vector
Contains a Vector of Vectors containing
UMLSCandidate objects
+UMLSPhrase(): Konstruktor
+getOrginalString(out org:String)
+getPosition(out position:integer)
+getRelevant(out Relevant:boolean)
+getPhrase(out phrase:Phrase)
+getNP(out out:String)
+getTypeOfPhrase(out out:String)
+getMatch(out out:Vector)
+getPOS(out:Vector)
+getCandidates(out out:Vector)
+print(out text:StringBuffer)
+toString()(out text:String)
```

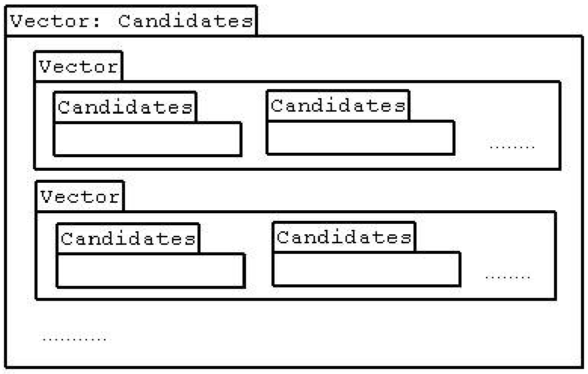


Figure 4-12 Class diagram of UMLSPhrase and schematic display of CUI and AUI vector

4.4.4 UMLSCandidate

The **UMLSCandidate** class stores the information about the concepts found for the phrase. It has ten different data fields, which identify and store additional information about the concept.

The class diagram in Figure 4-13 shows also the methods, which allow access to the information.

- String **getCui** ()
  It returns the concept id, which is the same as in the UMLS Metathesaurus.

- Vector **getAui** ()
  Each concept is composed of at least one atom. Some concepts have multiple atoms, because of different source vocabularies or synonyms. It returns a vector strings containing the atom id.

- Vector **getSemType** ()
  It returns a vector of strings containing the semantic types of the concept

- Vector **getSrcType** ()
  It returns a vector with the abbreviations of source vocabularies, where the matching atoms are found. The sources are abbreviated as in the UMLS. For example: “MSH” refers to the MeSH vocabulary.

- Vector **getTermType** ()
  It returns a vector with term types of the corresponding atoms. The term types are coded as in the UMLS. For example: “SY” means synonym, “MH” means preferred, “EN” means entry term, and so on.

- Vector **getSemTreeLabel** ()
  It returns a vector with labels for the concept determining the label’s position within the semantic tree.
  For Example: The label “A1.2.3.1” indicates that the concept comes from “Entity” (A), “Physical Object” (A1), “Anatomical Structure” (A1.2), “Fully Formed Anatomical Structure” (A1.2.3), where it is a child node of “Body Part, Organ, or Organ Component” (A1.2.3.1).

- Vector **getSemTypeID** ()
  It returns a vector of strings containing the semantic type ids of the concept.

- String **getCuiName** ()
  It returns a string with the preferred name of the Metathesaurus concepts stored in the CUIName variable.

- Vector **getSemGroup** ()
  It returns a vector with the semantic group names to which the semantic types of the concept are associated.

- Vector **getSemGroupID** ()
  It returns a vector with the ids of the semantic groups associated with the concept.

- Vector **getMatch** ()
  With this method one can determine to which string in the phrase the concept is connected.
  For example the concept name “Inspiration function” is linked to the string “inhaled” in the phrase “Inhaled steroids”.

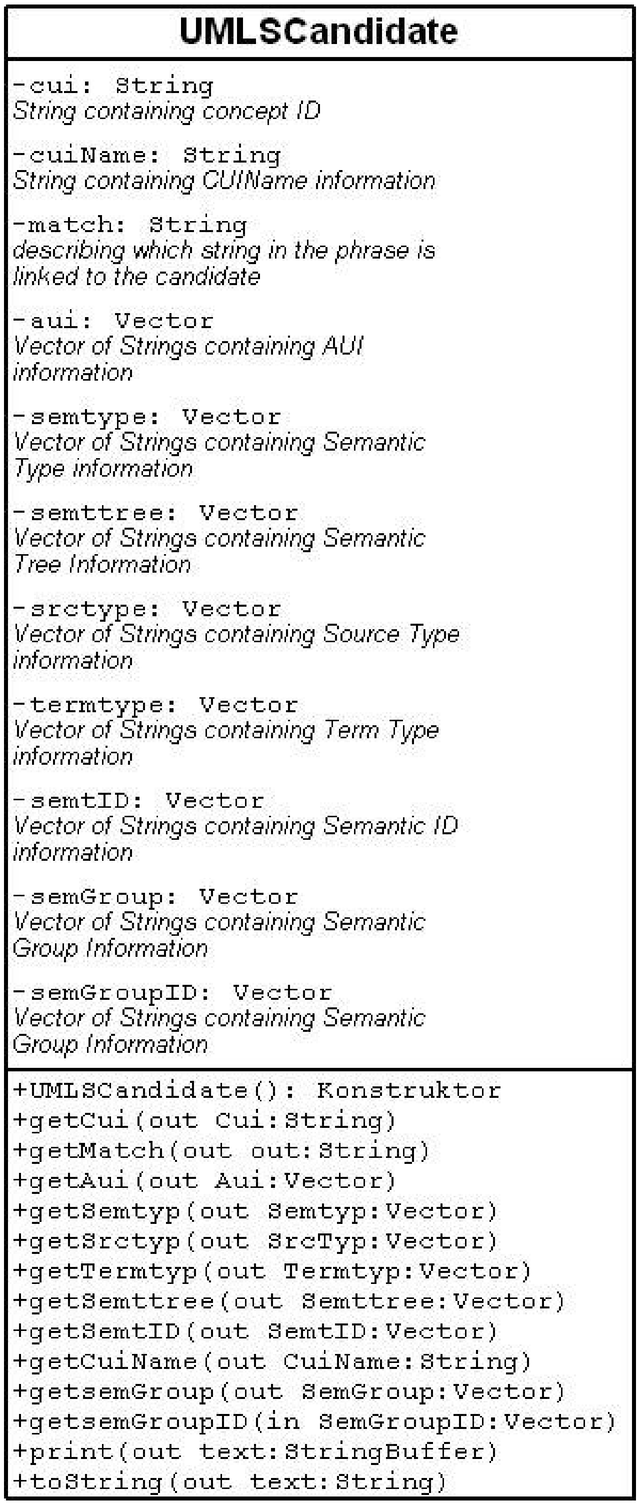

UMLSCandidate
-cui: String
String containing concept ID
-cuiName: String
String containing CUIName information
-match: String
describing which string in the phrase is linked to the candidate
-aui: Vector
Vector of Strings containing AUI information
-semtype: Vector
Vector of Strings containing Semantic Type information
-semttree: Vector
Vector of Strings containing Semantic Tree Information
-srctype: Vector
Vector of Strings containing Source Type information
-termtype: Vector
Vector of Strings containing Term Type information
-semtID: Vector
Vector of Strings containing Semantic ID information
-semGroup: Vector
Vector of Strings containing Semantic Group Information
-semGroupID: Vector
Vector of Strings containing Semantic Group Information
+UMLSCandidate(): Konstruktor
+getCui(out Cui:String)
+getMatch(out out:String)
+getAui(out Aui:Vector)
+getSemtyp(out Semtyp:Vector)
+getSrctyp(out SrcTyp:Vector)
+getTermtyp(out Termtyp:Vector)
+getSemttree(out Semttree:Vector)
+getSemtID(out SemtID:Vector)
+getCuiName(out CuiName:String)
+getsemGroup(out SemGroup:Vector)
+getsemGroupID(in SemGroupID:Vector)
+print(out text:StringBuffer)
+toString(out text:String)

Figure 4-13 Class diagram of UMLSCandidate.

4.4.5 Communication

The UMLSint class contains some complex methods, which I will explain a little more in detail. From this class the processing methods are invoked.

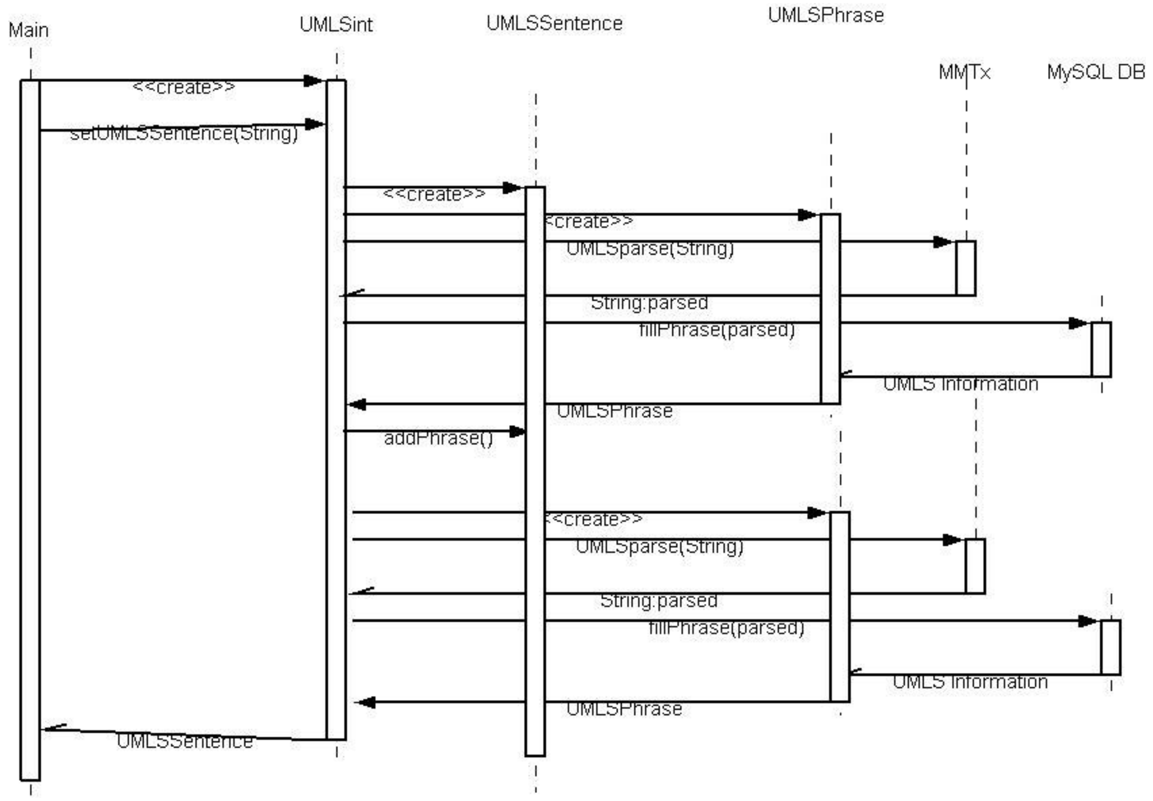


Figure 4-14 Sequence diagram of method „setUMLSSentence(String)"

UMLSSentence **setUMLSSentence** (String sentence)

This method is necessary to create and fill the UMLSSentence, UMLSPhrase, and UMLSCandidate objects as shown in Figure 4-14. The input is a simple string which represents a sentence. It is processed with the help of umlsParse which returns as many UMLSPhrase objects as can be found in the sentence. Also the UMLSPhrase holds UMLSCandidate objects, each representing one Metathesaurus concept. For each UMLSPhrase object the method fillPhrase is called. This method gathers the information together and fills the UMLSCandidate and UMLSPhrase objects with information. Afterwards all is put together into the UMLSSentence object and is returned to the invoker.

Vector **umlsParse** (String s)

This method is normally only called within the setUMLSSentence method. The input is a string, which normally represents a sentence or a part of a text. The main goal of the method is to parse the input into phrases with the help of MMTx. The call of MMTx is

```
Sentence aSentence = MMTx.processSentence(strSent);
```

This is all what is needed to parse the string. A little bit more complicating is how to get to the information. Due the nested nature of UMLS and therefore also MMTx, several loops are necessary to extract the information.

I will only show a short example to demonstrate the linking. For each string a sentence is created. It holds multiple phrases and for each phrase are multiple final mappings. And also every final mapping can hold multiple candidates or candidate combinations. And every candidate can have lots of atoms, source types and so on.

To reduce the complexity I created data fields in the UMLSPhrase class to store useful information. An important data field is the UMLSCandidate, which represents a Metathesaurus concept. After the process of parsing and identifying the sentence, an UMLSPhrase object is returned.

UMLSPhrase **fillPhrase** (UMLSPhrase up)

This method is normally only invoked in the combination with setUMLSSentence and umlsParse. With the information in the UMLSPhrase objects from umlsParse, it is now possible to search through the database of Metathesaurus and Semantic Network to fill the open data fields in the UMLSCandidate object. To achieve this it is necessary to use a module, which is able to connect to databases especially in this case to MySQL databases. I use the MySql Connector/J[4] which allows the connections. The connection is established with the information stored in the property file. I only connect in this method to three tables, “MRCONSO”, “MRSTY”, and “Semgroups”. In the MRCONSO table are information about concepts, concept IDs and their sources. In the MRSTY table are information about semantic types from concepts and semantic IDs. In the Semgroups table is the information about the semantic groups and its linkage stored.

All this information is read out and stored in the UMLSPhrase and UMLSCandidate objects, and afterwards the UMLSPhrase object is returned.

There are other methods which also connect to different tables and fulfil send queries to them. getRel, getSemRel, findRel, getDef, sendQueryMeta, and sendQuerySem are such methods.

sendQueryMeta and sendQuerySem methods are a little more special, because they can be used on any table to query any information. Only two parameters are needed, the SQL statement and the needed attributes. For example, the SQL statement

```
“select * from MRCONSO WHERE cui = ‘C0029882’;"
```

and the attributes “AUI”, “SAB”, and “TTY” will return all atoms from the concept “otitis media”, in which vocabulary source they are found, and what kind of term type they are.

It was necessary for each database to implement a separate method, because the connections are build differently.

4.5 Requirements and Limitations

As mentioned above the UMLSint package needs additional modules, which are the MMTx and the MySQL Connector/J. Those are the requirements to run the code. But there are also additional requirements, such as a Metathesaurus database and a Semantic Network database. It is not necessary that on every computer the databases are locally stored, but it is essential that on every system on which the UMLSint package runs MMTx and MySQL Connector/J are installed.

[4] http://www.mysql.com/products/connector/j/

There are also some needs on the databases. The Metathesaurus database is easy to construct with the help of MetaMorphoSYS and its loadfile. All tables and attributes are created the same way. We have seven tables and numerous attributes.

This looks different for the Semantic Network database. There is no loadfile available, therefore the loadfile from the Metathesaurus was choosen and the tables and attributes were manually altered, according to Figure 4-8. The UMLSint package will only work with the foregoing described database creation.

A big advantage of the UMLSint package is that it is not limited to a certain subject. Of course it is limited to the UMLS, but not to a certain area within it. It makes not difference if the database is mainly filled with cancer subjects, therapy, clinical devices, or drugs. The implementation will always work the same. So it is possible to enhance or manipulate the database on its own, without breaking the analyse process.

It is also possible to print out the gathered information. For example, the result for "vena caval stent filter" would look like this:

```
*************************************************
Original Sentence                : vena caval stent filter
Number of relevant Phrases found : 1
Postions of relevant Phrases     : 0
-------------------------------------------------
Position in Sentence : 0
Original Phrase      : vena caval stent filter
POS tags for phrase  : [noun, adj, noun, noun]
Np Phrase            : vena caval stent filter
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C0080306
  Concept Name       : Cava Filter, Vena
  Matching string    : Cava Filter, Vena
  Atom Identifier     : [A0131884, A0035991, A0059348, A0131883, A0035992, A0059353, A0073750,
A0059345, A0073749, A0059350]
  Source Type        : [MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH]
  Term Type          : [MH, PM, PM, PM, PM, PM, EP, EN, EN, EN]
  Semantic Type      : [Medical Device]
  Semantic Type ID   : [T074]
  Sem. Type Tree     : [A1.3.1]
  Semantic Group     : [Devices]
  Semantic Group ID  : [DEVI]

  Concept Identifier : C0038257
  Concept Name       : Stents
  Matching string    : Stent
  Atom Identifier    : [A0119502, A0119500]
  Source Type        : [MSH, MSH]
  Term Type          : [MH, PM]
  Semantic Type      : [Medical Device]
  Semantic Type ID   : [T074]
  Sem. Type Tree     : [A1.3.1]
  Semantic Group     : [Devices]
  Semantic Group ID  : [DEVI]
```

On the basis of this example it is possible to see a complex match. Two concepts can be found in one phrase and both are displayed.

5 Case Study

The purpose of this case study is, to determine how the UMLSint package works and what kinds of information are returned. It makes some kind of pre-interpretation of the input text by parsing and collecting information from the UMLS. The information can be used for a domain specific analysis of the text by IE.

For test purposes I used an extraction of an asthma guideline [43]. With the help of the UMLSintDemo class, the document can be automatically processed sentence by sentence. For demonstration purpose I use three sentences from the asthma guideline.

5.1 Example for a complex match

The first example is the sentence: "`Using two or more canisters of beta2 agonists per month or >10-12 puffs per day is a marker of poorly controlled asthma.`"

The sentence is parsed and mapped to Metathesaurus concept with the help of MMTx, it is invoked by *Sentence aSentence = MMTx.processSentence(strSent);*

MMTx now creates *phrase* objects from the input sentence:

Table 12 Parsed sentence number one

Using	two	canisters	of beta2 agonists	per month	>10-12 puffs	per day
a marker	of poorly controlled asthma.					

For each token of each phrase the POS tag is decided.

Table 13 Parsed sentence with POS tags

Using	two	canis-ters	of beta2 agonists	per month	>10-12 puffs	per day	a marker	of poorly con-trolled asthma.
[verb]	[noun]	[verb]	[prep, noun, noun]	[prep, noun]	[greaterThan, number, dash, number, noun]	[prep, noun]	[det, noun]	[prep, adv, adj, noun, pe-riod]

As we can see, the POS tag assignment is not always correct as seen with the example of *can-isters* which is not a verb. Nevertheless, the correct Metathesaurus concept is mapped as shown in Table 14.

Now the best mapped candidates must be extracted from the *phrase* object. For achievement, some methods have to be called. First, we have to get the list of all final mappings:

Then, for every *FinalMapping* object information of the candidates has to be extracted and stored in the *UMLSCandidate* object.

The extracted information consists of the preferred name of the Metathesaurus concept, the concept id and the string to which the concept was referred.

Table 14 Display of candidates for each phrase

	Candidate a			Candidate b		
	Concept ID	**Concept Name**	**Matched string**	**Concept ID**	**Concept Name**	**Matched string**
Using	C1524063	Using	Using			
two	C0205448	Two	Two			
canisters	C0336640	Canisters, device	Canister	C1556079	Entity Code – Canister	Canister
of beta2 agonists	C0243192	agonists	agonists			
per month	C0439231	month	month	C1561541	Precision – month	month
>10-12 puffs	C1533107	Puff	Puffs			
per day	C0439228	day	day	C1561539	Precision – day	day
a marker	C0005516	Biological markers	marker			
of poorly controlled asthma	C0205169	Bad	poorly			
	C0702113	controlled	controlled			
	C0004096	Asthma	Asthma			

In Table 14 the columns "Concept ID", "Concept Name", and "Matching Name" represent one candidate. In this example the phrases only have a maximum of two different matching candidates. In the last row a complex match is displayed. The phrase "of poorly controlled asthma" is a combination of three different concepts. In my first implementation with the help of the LEXICAL and TEXT tool, this phrase was not found and therefore no information was returned.

On the other hand, matching concepts are found with the help of the MMTx tool, because MMTx searches for every token and token combination, which links to a valid concept in the Metathesaurus. An internal rating chooses which concept or concept combination is most promising. How MMTX accomplish the creation of partial and complex matches is explained in details in Chapter 4-2-2 and Chapter 4-2-3.

After the identification of the matching concepts, additional information from the Metathesaurus and Semantic Network database is collected.

I will display the additional information for some parts of the sentence in Table 15.

Table 15 Detailed display of collected UMLS information

	Using	two	a marker	of poorly controlled asthma		
				poorly	controlled	asthma
Atom Identifier	[A7751422]	[A7164180]	[A0030908, A0290742, A0290759, A0083197, A0290735, A0030869, A0083195, A0290764]	[A6765507]	null	[A0027339, A7850156, A0027347, A0032834, A0027348, A0032835, A0027343]
Source Type	[MTH]	[MTH]	[MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH]	[MTH]	null	[MSH, MTH, MSH, MSH, MSH, MSH, MSH]
Term Type	[PN]	[PN]	[MH, EN, PM, EP, EN, EN, PM, PM]	[PN]	null	[MH, PN, PM, EP, PM, PM, EP]
Semantic Type	[Functional Concept]	[Quantitative Concept]	[Qualitative Concept]	[Qualitative Concept]	null	[Disease or Syndrome]
Semantic Type ID	[T169]	[T081]	[T080]	[T080]	null	[T047]
Sem. Type Tree ID	[A2.1.4]	[A2.1.3]	[A2.1.2]	[A2.1.2]	null	[B2.2.1.2.1]
Semantic Group	[Concepts & Ideas]	[Concepts & Ideas]	[Concepts & Ideas]	[Concepts & Ideas]	null	[Disorders]
Semantic Group ID	[CONC]	[CONC]	[CONC]	[CONC]	null	[DISO]

Not all matched concepts own additional information in the Metathesaurus or Semantic Network databases, as shown with the concept “controlled” in the sixth column of the Table 15. The reason why no additional information gets displayed is, because the concept is found in the SPECIALIST lexicon but the concept is not available in the Metathesaurus, as we use only a narrow set of sources, which do not contain the specified concept. In the last column of the Table 15 the concept “asthma”, is represented with multiple atom ids. These ids represent different spellings and synonyms as well as the origin from different source vocabularies. In general this is valid for all concepts with multiple atoms.

A complete output of this example is available in the Appendix A, Chapter sentence one.

5.2 Example for multiple matches

The second example is the sentence: "`Inhaled steroids are the recommended preventer drug for adults and children for achieving overall treatment goals.`"

The sentence is parsed and mapped to Metathesaurus concept with the help of MMTx, it is invoked by *Sentence aSentence = MMTx.processSentence(strSent);*

MMTx now creates *phrase* objects from the input sentence:

Table 16 Parsed sentence number two

Inhaled Steroids	the recommended preventer drug	for adults
children	for achieving overall treatment goals.	

For each token of each phrase the POS tag is decided.

Table 17 Parsed sentence with POS tags

Inhaled Steroids	the recommended preventer drug	for adults	children	for achieving overall treatment goals.
[adj, noun]	[det, adj, noun, noun]	[prep, noun]	[noun]	[prep, noun, adj, noun, noun, period]

Now the best mapped candidates must be extracted from the *phrase* object. For achievement, some methods have to be called. First, we have to get the list of all final mappings:

aPhrase.getBestFinalMappings();

Then, for every *FinalMapping* object information of the candidates has to be extracted and stored in the *UMLSCandidate* object.

FinalMappingt.getCandidates();

The extracted information consists of the preferred name of the Metathesaurus concept, the concept id and the string to which the concept was referred.

Table 18 Display of candidates for each phrase

	Candidate a			**Candidate b**		
	Concept ID	**Concept Name**	**Matched string**	**Concept ID**	**Concept Name**	**Matched string**
Inhaled steroids	C0004048	Inspiration function	Inhaled	C0004048	Inspiration function	Inhaled
	C0338671	Abuse of steroids	Steroids	C0038317	Steroids	Steroids
the recommended preventer drug	C1292733	Prevents	Prevents	C0309872	PREVENT	PREVENT
	C0013227	Pharmaceutical Preparations	Drug	C0013227	Pharmaceutical Preparations	Drug
for adults	C0001675	Adult	Adults			
children	C0008059	Child	Children	C1552465	Chronic Disease Hospital - Children	Children
for achieving overall treatment goals	C0282416	Overall Publication Type	Overall	C1561607	Overall	Overall
	C0679840	treatment goals	treatment goals	C0679840	treatment goals	treatment goals

In the Table 18 the columns “Concept ID”, “Concept Name”, and “Matching String” represent one candidate. The phrase can be composed of several concepts, which means “inhaled steroid” is matched to two different concepts, one for “inhaled” and one for “steroid”. Furthermore, in this example there are multiple concepts for steroids. Therefore, all combinations of inhaled and steroids are displayed.

The part "preventer" from the phrase "the recommended preventer drug" undergoes a morphological analysis in the MMTx. No attention is paid to upper or lower case. As a result, the matching string represents the morphological candidate, to which the concept from the Metathesaurus is matched.

In this case, "prevents" and "PREVENT" have the same distance value to "preventer" and therefore the found concepts have the same score in MMTx.

In this example the phrases only have a maximum of two different matching candidates, except of the phrase “children”. The phrase has got available a total of four different candidates, but due the limited space only two candidates will be displayed in Table 18.

After the identification of the matching concepts, additional information from the Metathesaurus and Semantic Network database is collected.

In Table 19 we show this information for selected parts of the sentence.

Table 19 Detailed display of collected UMLS information

	Inhaled steroids		for adults	for achieving overall treatment goals	
	Inhaled	Steroids		Overall	treatment goals
Atom Identifier	[A8554586, A2782463, A2783622, A0073986, A0073988, A2786166, A2786168, A2790552, A2790553, A2788531, A2788532, A2796166, A2790558]	[A0119651, A7186594]	[A0020365, A0020389]	[A8555297]	null
Source Type	[MTH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH]	[MSH, MTH]	[MSH, MSH]	[MTH]	null
Term Type	[PN, MH, PM, EP, PM, EN, PM, PM, PM, EN, PM, PM, PM]	[MH, PN]	[MH, EN]	[PN]	null
Semantic Type	[Organ or Tissue Function]	[Steroid]	[Age Group]	[Qualitative Concept]	null
Semantic Type ID	[T042]	[T110]	[T100]	[T080]	null
Sem. Type Tree ID	[B2.2.1.1.2]	[A1.4.1.2.1.9.1]	[A2.9.4]	[A2.1.2]	null
Semantic Group	[Physiology]	[Chemicals & Drugs]	[Living Beings]	[Concepts & Ideas]	null
Semantic Group ID	[PHYS]	[CHEM]	[LIVB]	[CONC]	null

Not all matched concepts own additional information in the Metathesaurus or Semantic Network databases, as shown with the concept “treatment goals” in the sixth column of the table. The reason why no additional information is displayed is because the concept is only found in the SPECIALIST lexicon. The concept “inhaled” in the second column, is represented with multiple atom ids. These ids represent different spellings and synonyms, as well as the origin from different source vocabularies.

A complete output of this example is available in the Appendix A, Chapter sentence two.

5.3 Example for ambiguous information

The third example is the sentence: "`Antihistamines and ketotifen are ineffective.`"

The sentence is parsed and mapped to Metathesaurus concept with the help of MMTx, it is invoked by *Sentence aSentence = MMTx.processSentence(strSent);*

MMTx now creates *phrase* objects from the input sentence:

Table 20 Parsed sentence number three

Antihistamines	ketotifen

For each token of each phrase the POS tag is decided.

Table 21 Parsed sentence with POS tags

Antihistamines	ketotifen
[noun]	[adv]

As we can see, the POS tag assignment is not always correct as seen with the example of *ketotifen* which is not a adverb. Nevertheless, the correct Metathesaurus concept is mapped.

Now the best mapped candidates must be extracted from the *phrase* object. For achievement, some methods have to be called. First, we have to get the list of all final mappings:

aPhrase.getBestFinalMappings();

Then, for every *FinalMapping* object information of the candidates has to be extracted and stored in the *UMLSCandidate* object.

FinalMappingt.getCandidates();

The extracted information consists of the preferred name of the Metathesaurus concept, the concept id and the string to which the concept was referred.

Table 22 Display of candidates for each phrase

	Candidate a			**Candidate b**		
	Concept ID	**Concept Name**	**Matched string**	**Concept ID**	**Concept Name**	**Matched string**
Antihistamines	C0003360	Antihistamines	Antihistamines			
ketotifen	C0022642	Ketotifen	Ketotifen			

In Table 22 the columns "Concept ID", "Concept Name", and "Matching Name" represent one candidate. In this example the phrases only have a maximum of one matching candidate.

After the identification of the matching concepts, additional information from the Metathesaurus and Semantic Network database are collected.

Table 23 Detailed display of collected UMLS information

	Antihistamines	Ketotifen
Atom Identifier	[A7755557, A0013968]	[A0077163, A0077162, A0077165, A0077166, A0077167, A0383931, A0526952]
Source Type	[MSH, MTH]	[RXNORM, MSH, MSH, MSH, MSH, MSH, MSH]
Term Type	[PEP, PN]	[IN, MH, EN, EN, EN, EN, N1]
Semantic Type	[Pharmacologic Substance]	[Organic Chemical, Pharmacologic Substance]
Semantic Type ID	[T121]	[T109, T121]
Sem. Type Tree ID	[A1.4.1.1.1]	[A1.4.1.2.1, A1.4.1.1.1]
Semantic Group	[Chemicals & Drugs]	[Chemicals & Drugs, Chemicals & Drugs]
Semantic Group ID	[CHEM]	[CHEM, CHEM]

The concept "Ketotifen" in the last column shows multiple entries for Semantic Type and other semantically related information. This means also that the assignment of semantic types is sometimes ambiguous and a disambiguation has to be accomplished by incorporating the concept's context within the text.

A complete output of this example is available in the Appendix A, Chapter sentence three.

6 Summary and Future Work

The amount of information published is progressively increasing, which makes it very hard to keep track without the help of computer systems. NLP technologies try to aid us by automatically analysing text and illustrating it in a condensed view. Part of the NLP technologies is IE.

A popular definition from Yangarber is that IE is "an emerging NLP technology whose function is to process unstructured, natural language text, to locate specific pieces of information, or facts in the text, and to use these facts to fill a database." [5]

IE systems are consisting of several components such as tokenization of the input, lexical and morphological processing, a syntactic analysis, a domain-specific processing, text sectionizing and filtering, Part of Speech tagging, coreferencing, and merging of partial results. Some of these components require some kind of knowledge base in the background to function correctly. Normally, the knowledge base may consist of dictionaries, vocabularies, thesauri, terminologies, or ontologies, or a combination of those.

There are many different terminologies and ontologies and nearly each of them is designed for a specific function, also in the medical domain. UMLS tries to display the knowledge by merging many different vocabularies and ontologies together. It stores the obtained information within three layers, the SPECIALIST Lexicon, the Metathesaurus, and the Semantic Network. All three combined together are used to display knowledge about medicine. Furthermore, UMLS provides a number of tools to analyse the lexical and morphological structure of text.

The objective of my designed package (UMLSint) is to retrieve the information provided by UMLS. I tried two different approaches to obtain the information.

The first approach uses the LEXICAL and TEXT Tools from UMLS to process, parse, analyse, and identify medical texts. The second approach uses MMTx which automatically maps the medical text to the UMLS database entries. Afterwards, the identified phrases are searched in the UMLS database and several data fields are extracted and returned.

The approach with the MMTx tool identified more phrases and was therefore the decision maker for choosing the second approach.

To use the UMLSint package it is necessary to establish a database for the UMLS Metathesaursus and the Semantic Network and install to the MMTx. For the communication with the database a database interface must be prepared in addition. UMLSint takes on the access and the communication with the database.

To enhance the functionality of the UMLSint package, a few things can be added in the future. At the moment the implementation works only on single sentences; it does not connect different sentences together. Also within the sentence and the phrase, it is difficult to clarify the relation among the elements. For future work it would be useful to enhance the detection of relationships within a sentence and between sentences. With this additional aid, even better results for IE could be possible.

7 Conclusions

The aim of this thesis is to create a plain interface to access the information provided by UMLS. With the help of several tools such as MetaMap Transfer (MMTx) it was possible to create the UMLSint package. The UMLSint package contains multiple functions. The target function is to provide the information from UMLS. However, to get this information different functions have to be combined.

The UMLSint package is designed to process texts with a medical background, for example medical guidelines, diagnosis, therapy schedules, instruction leaflets for drugs and so on. These texts need first to be analysed in a matter of linguistic views. The texts are tagged with information and logical elements are formed with the help of the SPECIALIST lexicon. This tagging on its own is already very useful for NLP systems.

With the logical elements, it is now possible to make a mapping to the UMLS Metathesaurus concepts with the help of MMTx. The Metathesaurus is also a reference book for medical terms with definitions and associations. Therefore, the found matches can be looked up and subject-specific words can be explained. The UMLSint package then provides all sorts of information, such as semantic type, source vocabulary, term type, semantic group, and so on.

It uses the MMTx tool to create and identify the logical elements. The main advantage of this approach is to receive good analysis of the logical elements and mappings to the Metathesaurus concepts. Hence, the program needs long processing time and the information is encapsulated in a complex data structures.

The UMLSint data structure now holds information from the UMLS provided by the MMTx tool. They are combined to one easy accessible and powerful tool.

In return this tool supplies results from lexical, morphological, and semantic knowledge. This information can be used by IE and NLP systems as a base for further text analysis.

During the work on my thesis, I figured out that the existing UMLS database is very large and searching takes a long time. Furthermore, the UMLS database is quite confusing and inscrutable. The complete information are set up in a huge amount of various attributes, some of them supply information with no clear purpose. Furthermore, the access for the MMTx tool is not easy due to its nested nature and the processing time takes long as well.

Appendix A

This Appendix shows the output of the three sample text and displays all found information with the help of the UMLSint print method.

Sentence number one

```
************************************************
Original Sentence                  : Using two or more canisters of beta2
agonists per month or >10-12 puffs per day is a marker of poorly controlled
asthma.
Number of relevant Phrases found : 9
Postions of relevant Phrases     : 0, 1, 2, 3, 4, 5, 6, 7, 8,
------------------------------------------------
Position in Sentence : 0
Original Phrase      : Using
POS tags for phrase  : [verb]
Np Phrase            : Using
Type of Phrase       : VERB_PHRASE

  Concept Identifier : C1524063
  Concept Name       : Using
  Matching string    : Using
  Atom Identifier    : [A7751422]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Functional Concept]
  Semantic Type ID   : [T169]
  Sem. Type Tree     : [A2.1.4]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

------------------------------------------------
Position in Sentence : 1
Original Phrase      : two
POS tags for phrase  : [noun]
Np Phrase            : two
Type of Phrase       : UNKNOWN_PHRASE

  Concept Identifier : C0205448
  Concept Name       : Two
  Matching string    : Two
  Atom Identifier    : [A7164180]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Quantitative Concept]
  Semantic Type ID   : [T081]
  Sem. Type Tree     : [A2.1.3]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

-------------------------------------------------
Position in Sentence : 2
Original Phrase      : canisters
POS tags for phrase  : [verb]
Np Phrase            : canisters
Type of Phrase       : VERB_PHRASE

  Concept Identifier : C0336640
  Concept Name       : Canister, device
  Matching string    : Canister
  Atom Identifier    : [A8572743]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Manufactured Object]
  Semantic Type ID   : [T073]
  Sem. Type Tree     : [A1.3]
  Semantic Group     : [Objects]
  Semantic Group ID  : [OBJC]

Position in Sentence : 2
Original Phrase      : canisters
POS tags for phrase  : [verb]
Np Phrase            : canisters
Type of Phrase       : VERB_PHRASE

  Concept Identifier : C1556079
  Concept Name       : Entity Code - Canister
  Matching string    : Canister
  Atom Identifier    : [A8572742]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Intellectual Product]
  Semantic Type ID   : [T170]
  Sem. Type Tree     : [A2.4]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

------------------------------------------------
Position in Sentence : 3
Original Phrase      : of beta2 agonists
POS tags for phrase  : [prep, noun, noun]
Np Phrase            : beta2 agonists
Type of Phrase       : OF_PREP_PHRASE

  Concept Identifier : C0243192
  Concept Name       : agonists
  Matching string    : agonists
  Atom Identifier    : [A3879792]
  Source Type        : [MSH]
  Term Type          : [TQ]
  Semantic Type      : [Pharmacologic Substance]
  Semantic Type ID   : [T121]
  Sem. Type Tree     : [A1.4.1.1.1]
  Semantic Group     : [Chemicals & Drugs]
  Semantic Group ID  : [CHEM]

------------------------------------------------
Position in Sentence : 4
Original Phrase      : per month
POS tags for phrase  : [prep, noun]
Np Phrase            : month
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C0439231
  Concept Name       : month
  Matching string    : month
  Atom Identifier    : [A8555162]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Temporal Concept]
  Semantic Type ID   : [T079]
  Sem. Type Tree     : [A2.1.1]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

Position in Sentence : 4
Original Phrase      : per month
```

```
POS tags for phrase  : [prep, noun]
Np Phrase            : month
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C1561541
  Concept Name       : Precision - month
  Matching string    : month
  Atom Identifier    : [A8555161]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Idea or Concept]
  Semantic Type ID   : [T078]
  Sem. Type Tree     : [A2.1]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

-------------------------------------------------
Position in Sentence : 5
Original Phrase      : >10-12 puffs
POS tags for phrase  : [greaterThan, number, dash, number, noun]
Np Phrase            : 10 12 puffs
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C1533107
  Concept Name       : Puff
  Matching string    : Puffs
  Atom Identifier    : []
  Source Type        : []
  Term Type          : []
  Semantic Type      : []
  Semantic Type ID   : []
  Sem. Type Tree     : []
  Semantic Group     : []
  Semantic Group ID  : []

-------------------------------------------------
Position in Sentence : 6
Original Phrase      : per day
POS tags for phrase  : [prep, noun]
Np Phrase            : day
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C0439228
  Concept Name       : day
  Matching string    : day
  Atom Identifier    : [A7751166]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Temporal Concept]
  Semantic Type ID   : [T079]
  Sem. Type Tree     : [A2.1.1]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

Position in Sentence : 6
Original Phrase      : per day
POS tags for phrase  : [prep, noun]
Np Phrase            : day
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C1561539
  Concept Name       : Precision - day
  Matching string    : day
  Atom Identifier    : [A8483906]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Idea or Concept]
  Semantic Type ID   : [T078]
  Sem. Type Tree     : [A2.1]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

-------------------------------------------------
Position in Sentence : 7
Original Phrase      : a marker
POS tags for phrase  : [det, noun]
Np Phrase            : marker
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C0005516
  Concept Name       : Biological Markers
  Matching string    : marker
  Atom Identifier     : [A0030908, A0290742, A0290759, A0083197, A0290735,
A0030869, A0083195, A0290764]
  Source Type        : [MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH]
  Term Type          : [MH, EN, PM, EP, EN, EN, PM, PM]
  Semantic Type      : [Qualitative Concept]
  Semantic Type ID   : [T080]
  Sem. Type Tree     : [A2.1.2]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

-------------------------------------------------
Position in Sentence : 8
Original Phrase      : of poorly controlled asthma
POS tags for phrase  : [prep, adv, adj, noun, period]
Np Phrase            : poorly controlled asthma
Type of Phrase       : OF_PREP_PHRASE

  Concept Identifier : C0205169
  Concept Name       : Bad
  Matching string    : Poorly
  Atom Identifier    : [A6765507]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Qualitative Concept]
  Semantic Type ID   : [T080]
  Sem. Type Tree     : [A2.1.2]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

  Concept Identifier : C0702113
  Concept Name       : Controlled
  Matching string    : Controlled
  Atom Identifier    : []
  Source Type        : []
  Term Type          : []
  Semantic Type      : []
  Semantic Type ID   : []
  Sem. Type Tree     : []
  Semantic Group     : []
  Semantic Group ID  : []

  Concept Identifier : C0004096
  Concept Name       : Asthma
  Matching string    : Asthma
  Atom Identifier     : [A0027339, A7850156, A0027347, A0032834, A0027348,
A0032835, A0027343]
  Source Type        : [MSH, MTH, MSH, MSH, MSH, MSH, MSH]
  Term Type          : [MH, PN, PM, EP, PM, PM, EP]
  Semantic Type      : [Disease or Syndrome]
  Semantic Type ID   : [T047]
  Sem. Type Tree     : [B2.2.1.2.1]
  Semantic Group     : [Disorders]
  Semantic Group ID  : [DISO]

--------------------------------------------------
**************************************************
```

Sentence number two

```
************************************************

Original Sentence                  : Inhaled steroids are the recommended
preventer drug for adults and children for achieving overall treatment
goals.

Number of relevant Phrases found : 5

Postions of relevant Phrases     : 0, 1, 2, 3, 4,

------------------------------------------------

Position in Sentence : 0

Original Phrase      : Inhaled steroids

POS tags for phrase  : [adj, noun]

Np Phrase            : Inhaled steroids

Type of Phrase       : NOUN_PHRASE


  Concept Identifier : C0004048

  Concept Name       : Inspiration function

  Matching string    : Inhaled

  Atom Identifier     : [A8554586, A2782463, A2783622, A0073986, A0073988,
A2786166,  A2786168,  A2790552,  A2790553,  A2788531,  A2788532,  A2796166,
A2790558]

  Source Type         : [MTH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH,
MSH, MSH, MSH]

  Term Type          : [PN, MH, PM, EP, PM, EN, PM, PM, PM, EN, PM, PM, PM]

  Semantic Type      : [Organ or Tissue Function]

  Semantic Type ID   : [T042]

  Sem. Type Tree     : [B2.2.1.1.2]

  Semantic Group     : [Physiology]

  Semantic Group ID  : [PHYS]


  Concept Identifier : C0338671

  Concept Name       : Abuse of steroids

  Matching string    : Steroids

  Atom Identifier    : [A7184333]

  Source Type        : [MTH]

  Term Type          : [PN]

  Semantic Type      : [Mental or Behavioral Dysfunction]

  Semantic Type ID   : [T048]

  Sem. Type Tree     : [B2.2.1.2.1.1]

  Semantic Group     : [Disorders]

  Semantic Group ID  : [DISO]


Position in Sentence : 0

Original Phrase      : Inhaled steroids

POS tags for phrase  : [adj, noun]
```

```
Np Phrase            : Inhaled steroids
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C0004048
  Concept Name       : Inspiration function
  Matching string    : Inhaled
  Atom Identifier     : [A8554586, A2782463, A2783622, A0073986, A0073988,
A2786166,  A2786168,  A2790552,  A2790553,  A2788531,  A2788532,  A2796166,
A2790558]
  Source Type         : [MTH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH, MSH,
MSH, MSH, MSH]
  Term Type          : [PN, MH, PM, EP, PM, EN, PM, PM, PM, EN, PM, PM, PM]
  Semantic Type      : [Organ or Tissue Function]
  Semantic Type ID   : [T042]
  Sem. Type Tree     : [B2.2.1.1.2]
  Semantic Group     : [Physiology]
  Semantic Group ID  : [PHYS]

  Concept Identifier : C0038317
  Concept Name       : Steroids
  Matching string    : Steroids
  Atom Identifier    : [A0119651, A7186594]
  Source Type        : [MSH, MTH]
  Term Type          : [MH, PN]
  Semantic Type      : [Steroid]
  Semantic Type ID   : [T110]
  Sem. Type Tree     : [A1.4.1.2.1.9.1]
  Semantic Group     : [Chemicals & Drugs]
  Semantic Group ID  : [CHEM]

-------------------------------------------------
Position in Sentence : 1
Original Phrase      : the recommended preventer drug
POS tags for phrase  : [det, adj, noun, noun]
Np Phrase            : recommended preventer drug
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C1292733
  Concept Name       : Prevents
  Matching string    : Prevents
  Atom Identifier    : []
  Source Type        : []
  Term Type          : []
  Semantic Type      : []
  Semantic Type ID   : []
  Sem. Type Tree     : []
  Semantic Group     : []
  Semantic Group ID  : []

  Concept Identifier : C0013227
  Concept Name       : Pharmaceutical Preparations
  Matching string    : Drug
  Atom Identifier     : [A1047492, A1047493, A1054854, A0051633, A1047490,
A1054853]
  Source Type        : [MSH, MTH, MSH, MSH, MSH, MSH]
  Term Type          : [MH, PN, EN, EP, EN, PM]
  Semantic Type      : [Pharmacologic Substance]
  Semantic Type ID   : [T121]
  Sem. Type Tree     : [A1.4.1.1.1]
  Semantic Group     : [Chemicals & Drugs]
  Semantic Group ID  : [CHEM]

Position in Sentence : 1
Original Phrase      : the recommended preventer drug
POS tags for phrase  : [det, adj, noun, noun]
Np Phrase            : recommended preventer drug
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C0309872
  Concept Name       : PREVENT
  Matching string    : PREVENT
  Atom Identifier    : []
  Source Type        : []
  Term Type          : []
  Semantic Type      : []
  Semantic Type ID   : []
  Sem. Type Tree     : []
  Semantic Group     : []
  Semantic Group ID  : []

  Concept Identifier : C0013227
  Concept Name       : Pharmaceutical Preparations
  Matching string    : Drug
  Atom Identifier     : [A1047492, A1047493, A1054854, A0051633, A1047490,
A1054853]
  Source Type        : [MSH, MTH, MSH, MSH, MSH, MSH]
  Term Type          : [MH, PN, EN, EP, EN, PM]
  Semantic Type      : [Pharmacologic Substance]
  Semantic Type ID   : [T121]
  Sem. Type Tree     : [A1.4.1.1.1]
  Semantic Group     : [Chemicals & Drugs]
  Semantic Group ID  : [CHEM]

-------------------------------------------------
Position in Sentence : 2
Original Phrase      : for adults
POS tags for phrase  : [prep, noun]
Np Phrase            : adults
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C0001675
  Concept Name       : Adult
  Matching string    : Adults
  Atom Identifier    : [A0020365, A0020389]
  Source Type        : [MSH, MSH]
  Term Type          : [MH, EN]
  Semantic Type      : [Age Group]
  Semantic Type ID   : [T100]
  Sem. Type Tree     : [A2.9.4]
  Semantic Group     : [Living Beings]
  Semantic Group ID  : [LIVB]

-------------------------------------------------
Position in Sentence : 3
Original Phrase      : children
POS tags for phrase  : [noun]
Np Phrase            : children
Type of Phrase       : UNKNOWN_PHRASE

  Concept Identifier : C0008059
  Concept Name       : Child
  Matching string    : Children
  Atom Identifier    : [A0037690, A7184490, A0037787, A0400685]
  Source Type        : [MSH, MTH, MSH, MTH]
  Term Type          : [MH, PN, EN, SY]
  Semantic Type      : [Age Group, Intellectual Product]
  Semantic Type ID   : [T100, T170]
  Sem. Type Tree     : [A2.9.4, A2.4]
  Semantic Group     : [Living Beings, Concepts & Ideas]
  Semantic Group ID  : [LIVB, CONC]
```

```
Position in Sentence : 3
Original Phrase      : children
POS tags for phrase  : [noun]
Np Phrase            : children
Type of Phrase       : UNKNOWN_PHRASE

  Concept Identifier : C1552465
  Concept Name       : Chronic Disease Hospital - Children
  Matching string    : Children
  Atom Identifier    : [A8572013]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type       : [Manufactured Object, Health Care Related Organiza-
tion]
  Semantic Type ID   : [T073, T093]
  Sem. Type Tree     : [A1.3, A2.7.1]
  Semantic Group     : [Objects, Organizations]
  Semantic Group ID  : [OBJC, ORGA]

Position in Sentence : 3
Original Phrase      : children
POS tags for phrase  : [noun]
Np Phrase            : children
Type of Phrase       : UNKNOWN_PHRASE

  Concept Identifier : C1552467
  Concept Name       : General Acute Care Hospital - Children
  Matching string    : Children
  Atom Identifier    : [A8572012]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type       : [Manufactured Object, Health Care Related Organiza-
tion]
  Semantic Type ID   : [T073, T093]
  Sem. Type Tree     : [A1.3, A2.7.1]
  Semantic Group     : [Objects, Organizations]
  Semantic Group ID  : [OBJC, ORGA]

Position in Sentence : 3
Original Phrase      : children
POS tags for phrase  : [noun]
Np Phrase            : children
Type of Phrase       : UNKNOWN_PHRASE
```

```
  Concept Identifier : C1552473
  Concept Name       : Rehabilitation Hospital - Children
  Matching string    : Children
  Atom Identifier    : [A8572011]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type       : [Manufactured Object, Health Care Related Organiza-
tion]
  Semantic Type ID   : [T073, T093]
  Sem. Type Tree     : [A1.3, A2.7.1]
  Semantic Group     : [Objects, Organizations]
  Semantic Group ID  : [OBJC, ORGA]

-------------------------------------------------
Position in Sentence : 4
Original Phrase      : for achieving overall treatment goals
POS tags for phrase  : [prep, noun, adj, noun, noun, period]
Np Phrase            : achieving overall treatment goals
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C0282416
  Concept Name       : Overall Publication Type
  Matching string    : Overall
  Atom Identifier    : [A8555298, A0573065, A0573063, A0573064]
  Source Type        : [MTH, MSH, MSH, MSH]
  Term Type          : [PN, MH, EN, EN]
  Semantic Type      : [Intellectual Product]
  Semantic Type ID   : [T170]
  Sem. Type Tree     : [A2.4]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

  Concept Identifier : C0679840
  Concept Name       : treatment goals
  Matching string    : treatment goals
  Atom Identifier    : []
  Source Type        : []
  Term Type          : []
  Semantic Type      : []
  Semantic Type ID   : []
  Sem. Type Tree     : []
  Semantic Group     : []
  Semantic Group ID  : []
```

```
Position in Sentence : 4
Original Phrase      : for achieving overall treatment goals
POS tags for phrase  : [prep, noun, adj, noun, noun, period]
Np Phrase            : achieving overall treatment goals
Type of Phrase       : PREP_PHRASE

  Concept Identifier : C1561607
  Concept Name       : Overall
  Matching string    : Overall
  Atom Identifier    : [A8555297]
  Source Type        : [MTH]
  Term Type          : [PN]
  Semantic Type      : [Qualitative Concept]
  Semantic Type ID   : [T080]
  Sem. Type Tree     : [A2.1.2]
  Semantic Group     : [Concepts & Ideas]
  Semantic Group ID  : [CONC]

  Concept Identifier : C0679840
  Concept Name       : treatment goals
  Matching string    : treatment goals
  Atom Identifier    : []
  Source Type        : []
  Term Type          : []
  Semantic Type      : []
  Semantic Type ID   : []
  Sem. Type Tree     : []
  Semantic Group     : []
  Semantic Group ID  : []

-------------------------------------------------
*************************************************
```

Sentence number three

```
*************************************************
Original Sentence                  : Antihistamines and ketotifen are inef-
fective.
Number of relevant Phrases found : 2
Postions of relevant Phrases     : 0, 1,
-------------------------------------------------
Position in Sentence : 0
Original Phrase      : Antihistamines
POS tags for phrase  : [noun]
Np Phrase            : Antihistamines
Type of Phrase       : NOUN_PHRASE

  Concept Identifier : C0003360
  Concept Name       : Antihistamines
  Matching string    : Antihistamines
  Atom Identifier    : [A7755557, A0013968]
  Source Type        : [MSH, MTH]
  Term Type          : [PEP, PN]
  Semantic Type      : [Pharmacologic Substance]
  Semantic Type ID   : [T121]
  Sem. Type Tree     : [A1.4.1.1.1]
  Semantic Group     : [Chemicals & Drugs]
  Semantic Group ID  : [CHEM]

-------------------------------------------------
Position in Sentence : 1
Original Phrase      : ketotifen
POS tags for phrase  : [adv]
Np Phrase            : ketotifen
Type of Phrase       : ADVERB_PHRASE

  Concept Identifier : C0022642
  Concept Name       : Ketotifen
  Matching string    : Ketotifen
  Atom Identifier     : [A0077163, A0077162, A0077165, A0077166, A0077167,
A0383931, A0526952]
  Source Type        : [RXNORM, MSH, MSH, MSH, MSH, MSH, MSH]
  Term Type          : [IN, MH, EN, EN, EN, EN, N1]
  Semantic Type      : [Organic Chemical, Pharmacologic Substance]
  Semantic Type ID   : [T109, T121]
  Sem. Type Tree     : [A1.4.1.2.1, A1.4.1.1.1]
  Semantic Group     : [Chemicals & Drugs, Chemicals & Drugs]
  Semantic Group ID  : [CHEM, CHEM]

-------------------------------------------------
*************************************************
```

List of Figures

List of Tables

References

1. K. Riedling. Die Publikationsdatenbank der Technischen Universität Wien. *Mitteilungen der VÖB*, 58(2): 30—49, 2005.
2. FIZ Technik e.V. Fachinformation Technik, Effiziente Informationsversorgung. *Service-Katalog*: 6—7, 2006.
3. D.J.d.S. Price. *Little Science, Big Science*. Suhrkamp Verlag, 1974.
4. D.E. Appelt and D.J. Israel. Introduction to Information Extraction. *A Tutorial prepared for IJCAI-99*, 1999.
5. R. Yangarber, *Scenario Customization for Information Extraction*. 2001, New York University: New York.
6. M. Field and K. Lohr. Guidelines for clinical practice: from development to use. . *D.C: National Academy Press*, 1992.
7. T. Hapke. *Informationsflut und Informationsmangel*. 2004 [cited 6.07.2006]; Available from: http://www.tu-harburg.de/b/hapke/infolit/orient1.htm.
8. National Guideline Clearinghouse. *National Guideline Clearinghouse*. 2006 18.12.2006 [cited 2006 23.12]; Available from: http://www.guideline.gov/.
9. W. Leinfellner, E. Kraemer, and J.S. (eds.). *Language and Ontology*. in *Proceedings of the Sixth International Wittgenstein Symposium*. 1982. Kirchberg am Wechsel (Austria): Wien, Hölder-Pichler-Tempsky.
10. T.R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2): 199—220, 1993.
11. National Library of Medicine, *Medical Subject Headings*. updated annually, The Library: Bethesda, MD.
12. R. Cote, D. Rothwell, J. Palotay, R. Beckett, and L. Brochu. *The Systematized Nomenclature of Medicine*. SNOMED International, 1993.
13. World Health Organisation. *International Classification of Diseases (ICD)*. 2006 2006 [cited 2006 23.12]; Available from: http://www.who.int/classifications/icd/en/.
14. Unified Medical Language System®. *Unified Medical Language System (UMLS)*. 01.11.2006 [cited 24.04.2006]; Available from: http://www.nlm.nih.gov/research/umls/.
15. B.L. Humphreys and P.L. Schuyler. The Unified Medical Language System: Moving beyond the vocabulary of bibliographic retrieval. In N.C. Broering, Editor. *High- Performance Medical Libraries: advanced information management for the virtual era*, Meckler, Westport (CT), 1993, pages 31—44.
16. C. Tilley and J. Willis, *The Unified Medical Language System, What is it and how to use it?*, in *Learning Resources for the UMLS*. 2004.
17. J.R. Anderson and G.H. Bower. *Human Associative Memory*. John Wiley and Sons, New York, 1973.
18. B. Sundheim. *Overview of the Fourth Message Understanding Evaluation and Conference*. in *Proceedings of the Fourth Message Understanding Conference*. 1992. pp 3—21.

19. W. Lehnert, C. Cardie, D. Fisher, J. McCarthy, E. Riloff, and S. Soderland. Evaluating an {I}nformation {E}xtraction System. *Journal of Integrated Computer-Aided Engineering*, 1(6): 453—472, 1994.

20. Natural Language Processing Group. *GATE Information Extraction*. [cited; Available from: http://gate.ac.uk/ie/.

21. C.J.v. Rijesbergen. *Information Retrival*. 2nd Edition ed. Butterworths, London, 1979.

22. R. Gaizauskas and A.M. Robertson. *Coupling Information Retrieval and Information Extraction: A New Text Technology for Gathering Information from the Web*. in *Proc. of RIAO'97: Computer-Assisted Information Searching on the Internet*. 1997. Montreal, CA.

23. National Institute of Standards and Technology. *SAIC Information Extraction*. 2001 8.03.05 [cited 2007 10.1]; Available from: http://www-nlpir.nist.gov/related_projects/muc/index.html.

24. R. Grishman and B. Sundheim. Message Understanding Conference - 6: A Brief History. *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Kopenhagen, 1996, pages 466—471.

25. K. Kaiser, *VU Informationsextraktion aus Texten*. 2005, Information Engineering Group: Vienna.

26. D.E. Appelt. Introduction to Information Extraction. *AI Communications*, 12: 161—172, 1999.

27. O. Bodenreider. Medical Ontology Research. Lister Hill National Center for Biomedical Communications, 2001.

28. A.C. Browne, A.T. McCray, and S. Srinivasan. The Specialist Lexicon. Lister Hill National Center for Biomedical Communications, National Library of Medicine, 2000.

29. A.R. Aronson. *Effective Mapping of Biomedical Text to the UMLS Metathesaurus: The MetaMap Programm*. in *AMIA*. 2001.

30. SPECIALIST Text Tools. *Phrase Tokenizer - Parser*. 11.11.2006 [cited 11.03.2007]; Available from: http://lexsrv3.nlm.nih.gov/SPECIALIST/Projects/textTools/current/Usages/Parser.html.

31. A.R. Aronson, T.C. Rindflesch, and A.C. Browne. *Exploiting a Large Thesaurus for Information Retrieval*. in *Proc. of RIAO'94*. 1994.

32. National Library of Medicine. *Semantic Knowledge Representation (SKR)*. 16.02.2007 [cited 11.03.2007]; Available from: http://skr.nlm.nih.gov/papers/index.shtml#MetaMap.

33. A.T. McCray, S. Srinivasan, and A.C. Browne. *Lexical methods for managing variation in biomedical terminologies*. in *Proc. of the 18th SCAMC*. 1994.

34. D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. *A practical part-of-speech tagger*. in *Proc. of the Third Conference on Applied Natural Language Processing*. 1992.

35. A.R. Aronson. MetaMap Variant Generation. Technical Report, Lister Hill National Center for Biomedical Communications, National Library of Medicine, 2001.

36. A.R. Aronson. MetaMap Candidate Retrieval. Technical Report, Lister Hill National Center for Biomedical Communications, National Library of Medicine, 2001.

37. A.R. Aronson. MetaMap Evaluation. Technical Report, Lister Hill National Center for Biomedical Communications, National Library of Medicine, 2001.

38. A.R. Aronson. MetaMap Mapping Algorithm. Technical Report, Lister Hill National Center for Biomedical Communications, National Library of Medicine, 2001.

39. A.R. Aronson, *MetaMap: Mapping Text to the UMLS Metathesaurus*. 2006.

40. O. Bodenreider and A.T. McCray. Exploring semantic groups through visual approaches. *Biomedical Informatics*, 36(6): 414—432, 2003.

41. A.T. McCray, A. Burgun, and O. Bodenreider. *Aggregating UMLS semantic types for reducing conceptual complexity*. in *Medinfo*. 2001.

42. Z. Chen, Y. Perl, M. Halper, J. Geller, and H. Gu. Partitioning the UMLS Semantic Network. *IEEE Transactions on Information Technology in Biomedicine*, 6(2): 102—108, June 2002.

43. Scottish Intercollegiate Guidelines Network (SIGN) and British Thoracic Society. British Guideline on the Management of Asthma. A Clinical National Guideline. Scottish Intercollegiate Guidelines Network (SIGN), November 2005.