„Dunarea de Jos” University of Galati
Doctoral School of Fundamental and Engineering Sciences

UNIVERSITAS GALATIENSIS

DOCTORAL THESYS

- SUMMARY -

Contributions Regarding the Representation of Contextual Knowledge in Natural Language Processing

PhD Student
Cristian Niculiță

Scientific advisor	Prof. univ. dr. ing. Luminița DUMITRIU
Scientific committee	Conf. univ. dr. Corina FORĂSCU CSII dr. Verginica BARBU MITITELU Conf. univ. dr. ing. Emilia PECHEANU

Series I2: Computers and Information Technology No 6
GALATI
2020

Seriile tezelor de doctorat susținute public în UDJG începând cu 1 octombrie 2013 sunt:

Domeniul fundamental ȘTIINȚE INGINEREȘTI

Seria I 1:	**Biotehnologii**
Seria I 2:	**Calculatoare și tehnologia informației**
Seria I 3:	**Inginerie electrică**
Seria I 4:	**Inginerie industrială**
Seria I 5:	**Ingineria materialelor**
Seria I 6:	**Inginerie mecanică**
Seria I 7:	**Ingineria produselor alimentare**
Seria I 8:	**Ingineria sistemelor**
Seria I 9:	**Inginerie și management în agricultură și dezvoltare rurală**

Domeniul fundamental ȘTIINȚE SOCIALE

Seria E 1:	**Economie**
Seria E 2:	**Management**
Seria SSEF:	**Știința sportului și educației fizice**

Domeniul fundamental ȘTIINȚE UMANISTE ȘI ARTE

Seria U 1:	**Filologie- Engleză**
Seria U 2:	**Filologie- Română**
Seria U 3:	**Istorie**
Seria U 4:	**Filologie - Franceză**

Domeniul fundamental MATEMATICĂ ȘI ȘTIINȚE ALE NATURII

Seria C:	**Chimie**

Domeniul fundamental ȘTIINȚE BIOLOGICE ȘI BIOMEDICALE

Seria M:	**Medicină**

I give grateful thanks to Professor PhD. Eng. Luminița Dumitriu, who steered my ideas in the right direction and whose experience was essential for the completion of this doctoral thesis.

I thank the members of the thesis evaluation and presentation committee, whose valuable observations helped me polish the parts in need of more clarity.

I also thank all the colleagues who always supported me full of understanding.

Finally, from the bottom of my heart, a big thank you to my family for the unwavering trust, patience and encouragement they have given me all this time.

Table of contents

Introduction

The study presented in this doctoral dissertation stems from the fascination that whole generations of researchers in the field of computer science have shown on the understanding of natural language and from which we too have had virtually no chance to escape, thus leaving us taken and carried away by the wave of enthusiasm to study this vast field.

From our perspective, the main objective of studying natural language is to accumulate a sufficient ability to understand the human intuitive method of analysis, so as to make it possible to apply this knowledge in a practical way allowing natural interaction with man-made devices. However, despite the more or less important successes of infusing life to machine, the ultimate goal, obtaining a real artificial intelligence that can compete with the human way of analysis, still remains inaccessible even today.

A metaphorical way of looking at all these studies is that every researcher, analyzing a certain singular aspect, grabbed and pulled at the same time a virtual thread of a complex and yet impenetrable informational fabric, attempting to dismantle it, in hope to, ultimately, gain the reward hidden behind it, which is the very essence of how to understand the natural language.

In order to contribute to the general effort in the field of natural language processing, in this paper we have chosen to "pull" a little on the thread dedicated to the definition analysis. The main purpose was to be able to identify the definitions in the text in order to extract from them the essential information related to the concepts they describe. Also, a notable aspect regarding the definitional sentences we analyzed in this dissertation, is that they are usually important enough to provide by themselves a clear overview of the context in which they reside and thus we consider that they possess a relatively high summarizing capacity.

Of course, the detection and extraction of definitions has been studied previously and many researchers have proposed different practical methods. If the simplest among them resort to identifying definitions using manually defined lexico-syntactic patterns, the most complex use advanced machine learning techniques involving the use of neural networks and sophisticated ways of encoding the structure and lexico-syntactic features that are definition specific.

From the point of view of a researcher who aims to study the definition detection in the context of Romanian language, all these methods have a great disadvantage, namely that they were created for other languages, most of them addressing the English language. As far as we know, a specific method dedicated to this purpose has not been yet created and implemented for the Romanian language, which is why we set out to develop one that takes into account the particularities of the our language.

Although we initially started from the premise of borrowing an English method in order to slightly adapt it to our needs, we found along the way that large differences in complexity require the development of specific, much more advanced processing mechanisms, this constituting the most important contribution of this work.

Thesis Overview

Chapter 1 presents an overview of the current state of the art in terms of definition detection/extraction. Section 1.2 discusses a number of general notions that help crystallizing the notion of definition.

Section 1.3 makes a short presentation of preprocessing methods and preprocessing tools and section 1.4 discusses how the analysis of definitions was approached both theoretically but especially in practice.

Chapter 2 briefly presents the English method from which we started, which is based on lexico-syntactic patterns, as well as the process of adapting the training corpus used by it for the Romanian language.

To do this, a series of syntax changes were made (section 2.3), which proved necessary considering the increased complexity of the Romanian language. Section 2.4 introduces the concept of **linking words** which is a defining aspect of the Romanian method, and section 2.5 defines a series of measures describing their particularities.

Chapter 3 presents a study concerning the tools for tagging the parts of speech and for chunking, which are available for the Romanian language.

Also, section 3.2 presents a number of adaptations and improvements to the chosen tagging/chunking tool.

Chapter 4 introduces the concept of **simplification pattern**, element that plays an important role in obtaining the canonical form of the analyzed sentence, based on which a **simple pattern** is created, that will be used for definition detection.

To represent these simplification patterns, a relatively simple special language has been defined whose syntax is centered on the pattern component. Sections 4.2 and 4.3 present general aspects about the configuration file syntax of the simplification patterns, and section 4.5 describes in detail all the syntactic elements associated with individual pattern components.

Chapter 5 presents the process of creating the simple patterns.

Section 5.2 describes the internal pattern representation of the analyzed sentence during the simplification process.

Section 5.4 presents a number of pre-processing elements of the initial sentence. They apply a temporary simplification helping the tagging tool to associate the correct parts of speech to certain difficult terms.

Starting with section 5.5, the steps for obtaining the simple pattern are presented.

Section 5.14 describes in general terms the final simplification phase of noun phrases, which has two parts: the enumeration analysis phase and the simplification phase itself.

In the first phase, the analysis (section 5.15) aims to detect enumerations that may occur within noun phrases and which may be: noun, adjective or participle-type adjective enumerations. In the process of enumeration detection, a whole range of possible relational combinations is generated between terms, that are ranked based on reliability scores: **similarity**, **structure**, **discrepancy** and **distance** score.

Section 5.16 presents in detail the simplification process using the information gathered in the enumeration analysis phase.

At the end of the chapter, in section 5.17, it is described the procedure for storing the simple patterns, which is done by dividing them into sub-patterns corresponding to the formal units of the definition.

In Chapter 6, the experimental validation of the classifier is performed . Additionally, we study the hypothesis according to which the important **definitional concepts** in the document also have a high relative frequency. We must specify that the concepts mentioned above, correspond to the defined term and its super-concept of a definition.

Section 6.2 presents the classification results by calculating the usual evaluation measures of the classifiers: accuracy, recall, accuracy, F-measure, while section 6.3 analyzes the causes of classification errors.

Section 6.4 discusses the correlation between frequency (TF-IDF) and the importance of definitional concepts, describing two algorithmic variants to select these concepts, the validation of the results being done by human analysis.

The last chapter presents the final conclusions, the contributions of this thesis, as well as the possible directions for future research.

Images

Tables

Listings

1 Current status in the field of definition detection and extraction

1.1 Introduction

The acquisition of knowledge has been a major goal of natural language processing since the beginning. The general goal is to help the computers to read a text and express the knowledge it contains using a formal representation, suitable for answering questions and solving problems. However, progress has been difficult. The oldest approaches were manual, but the massive effort to encode the knowledge made them very expensive and limited to well-defined areas. Subsequently, the inclusion and development of machine learning-based approaches has substantially reduced this effort. Thus, important steps have been made towards a sufficiently automated solution that can take the process from one end to the other. Even so, it is important to note that supervised learning requires labeled data, which is itself a rather expensive activity and may even prove impossible for the knowledge acquisition on a large scale in areas that are open ended [1].

The nature of the text that is analyzed for definitional constructions detection has a major influence in the success of this action. In well-structured texts, such as technical or medical texts, automatic identification of definitions is possible even through the use of the structure cues and, maybe, of keywords [2]. For example, in most math textbooks, definitions are evidently marked in the text and usually have a special format. In contrast, in less structured texts, the identification of definitional constructions is generally much more difficult, as they are usually expressed in a more linguistically free form. In such cases, even a person has difficulty identifying definitions, and the process is laborious, requiring careful consideration of the entire text. Unfortunately, the detection methods developed so far only manage to automate this process in a rather inefficient way for the less structured texts.

Formally, the extraction of definitions aims to detect term-definition pairs from a plain text. Depending on the desired level of detail we can only intend to detect the definitions, which only involves a general classification of sentences into definitions and non-definitions or we may want to accurately identify the components of a definition, in which case a more complex labeling operation is required. [3].

Also, the extraction of definitions is closely related to the identification of concept – super-concept (hyponym - hyperonym) relationships. In many cases, the methods of definition extraction are also able to identify these pairs, because they focus on detecting definitions that largely conform to the Aristotelian pattern, known from classical literature. Having a well-defined structure, this pattern allows for the definitions identification predominantly using lexico-syntactic methods.

As mentioned earlier, manually identifying definitions in a large text, even one with a structured nature, requires a long effort and should ideally be automated. Given the fact that for the Romanian language there is no method for definition detection, in this dissertation, we will aim to develop such a method that uses lexico-syntactic patterns, and is based on the method described by Navigli and Velardi in [4]. We must specify that along with the method we also borrowed the training corpus used by it, which contains definitions in the classical Aristotelian format, which will be described in the next section.

1.2 General notions related to definitions and glossaries

The main purpose of a definition is to explicitly establish the meaning of a term used in a technical or scientific document [5], thus providing information that could be useful in various situations.

A first step in detecting and extracting definitions from corpora must be their very definition [6], in order to clearly know what we are looking for and what is the form in which the information of interest is found in the text. The definitions typologies that have been frequently mentioned in the literature [7], [8] can be grouped into three categories: formal, semi-formal and non-formal definition.

The formal definition corresponds to the Aristotelian definitional type, described in [9] by the equation X = Y + C. In this, X represents *definiendum* (defined concept, hyponym). The Y + C ensemble, called *definiens*, is the explanation given for the defined term. Y is the generic class, *genus* (super-concept, hyperonym), to which X belongs, and C represents the specific characteristics, *diferentia*, which describes how X is different from the other elements of Y. The equality sign establishes the equivalence relationship between these elements. In the text it the expression of the definitional verb called *definitor*, which plays a central role in most methods of definition detection/extraction. Because we will use the term hyperonymy quite frequently, we will provide a definition of it:

> ***Hyperonymy*** is a semantic relationship of superordination between meanings of words.

In our case, we will use the term hyperonymy to refer to the relationship between the defined concept and its super-concept.

A semi-formal definition describes the definiendum only by its specific characteristics or attributes [10]. Formal and semi-formal definitions can be simple (expressed in a single sentence), or complex (two or more sentences).

A non-formal definition tries to have an ordinary form, so that the reader can see the familiar element of the new term [7]. This can be the association with a synonym, a paraphrase or a grammatical derivation.

The common point for all these visions regarding the same linguistic object, referred to by the common name of *definition*, is that they all have the same didactic purpose of disambiguating the meaning of a certain lexical element. These descriptions see the definition as the association between a term and its hyperonym, or between a certain term and its characteristics.

Dedicated studies dealing with definition typologies also mention additional ways of expressing them. A unified typology is proposed in [11], out of which we present, as being of interest, the following three new categories:

- definitions expressed by "low level" linguistic markers, including punctuation marks such as parentheses, quotation marks, hyphens, colons;
- definitions expressed by lexical markers: linguistic or metalinguistic lexical elements;
- definitions expressed by "high-level" linguistic markers: syntactic patterns, such as anaphora or apposition.

When complete, definitions generally have a regular structure, containing a number of lexical and metalinguistic patterns that can be automatically recognized [12]. The context of a definition is a specialized fragment of a text that is, in principle, structured around a term and its definition, both elements being connected by typographic or lexico-syntactic patterns [13]. The typographic patterns are mainly expressed by punctuation marks (commas, parentheses), while lexico-syntactic patterns include definitional verbs such as: "*to define*" or "*to signify*", as well as discursive markings, for example "*i.e.*" or "*in other words*". In addition, definitional contexts may also include pragmatic models, like "*in general terms*", or "*in this sense*", which state the conditions of use for the term or clarify its meaning.

Following the analysis of the linguistic forms that the definitional constructions can have, we can identify these specific patterns, which can then be used within automatic searching tools. This approach has been implemented with varying degrees of success. It should be noted that the results obtained on technical texts are generally better than in the case of non-technical texts, where the level is just satisfactory. Two factors limit the success of these results. On the one hand, the relative importance of different linguistic forms is difficult to assess, and thus is often ignored. On the other hand, finding effective language forms that are able to recognize most real definitions but do not accept non-definitions requires time and expertise and has proved an extremely difficult endeavor [2].

The process of creating glossaries is implicitly dependent on the specialized knowledge of the given field, its concepts and language. In case of constantly evolving scientific fields, glossaries need to be regularly updated and expanded. Creating specialized glossaries manually is therefore expensive. Alternatively, automatic and semi-automatic methods have been proposed.

In [14] two roles of specialized glossaries are identified. Firstly, they represent linguistic resources that summarize the important terms of a specialized field and secondly, they are knowledge resources, by providing definitions for the concepts those terms represent. Glossaries have an obvious utility as reference sources. A study on the use of lexicographic aids in specialized translation showed that glossaries are among the top five resources used [15]. They have also been shown to facilitate the comprehension of texts and the acquisition of knowledge while studying [16]. Grouping definitions in a glossary makes it easy to find information about keywords quickly, as well as the context in which they can be found, improving the knowledge assimilation [2].

From the computer processing point of view, electronic glossaries have proved to be valuable resources in certain natural language processing activities such as question answering [17], meaning disambiguation [18], ontology learning [19].

1.3 Text categories and preprocessing tools

The training and evaluation of the methods for extracting the definitions and hyperonymy relations is done using various collections of texts. To obtain optimal results, in the initial phase of creating patterns or training classification/clustering algorithms, carefully selected and, in most cases, manually annotated corpora are used. Due to the difficulty of obtaining them, they tend to be small in size. On the other hand, in the testing phase, the used corpora are as large as possible to alleviate the problem of reduced data coverage.

The **text structure** gives rise to two categories: specialized/technical corpora that generally have a high degree of structuring [6], [13], [14] and heterogeneous corpora created from texts collected from the internet from various sources [20], [21], [22], [23], [24], [25]. A commonly used source of texts, that is somewhere in the middle, is Wikipedia, where the information has to some extent a predictable format.

At one end of the spectrum there are methods for definition extraction that generally require well-structured text collections to achieve good results [6], [13]. For example, the success of the DEFINDER system [26], registering values of 0.87 for precision and 0.75 for the recall, is largely due to the well-structured medical corpus that was used. On the other hand, there are clustering approaches for creating taxonomies that can use heterogeneous texts [20], [21].

From the point of view of the **adaptability of the method** for which they are used, corpora can be specific to a certain language [6], [13], [2], extensible to other languages [21], [22], [24],

[27], or are multilingual from the beginning [23]. Usually the methods that are largely based on classical patterns are more difficult to adapt because the lexico-syntactic structuring differs from one language to another.

In general, texts are analyzed at least with the help of a tagger that labels the speech parts and possibly with a semantic analyzer that creates a dependency tree. This information is used both for creating lexico-syntactic patterns and for devising features used for clustering or classification. Some of the most commonly used tools are: Minipar [20], [21], Stanford POS Tagger [2], [25], Stanford Parser [24], Tree Tagger [4], TnT Tagger [14], etc.

1.4 Summary of definition extraction approaches

The study of the automatic extraction of definition knowledge was approached both from a theoretical-descriptive and from a practical perspective.

One of the first **theoretical-descriptive studies** is [12], which describes the contextual environment in which the terms appear. Here it is mentioned that, when authors define a term, they use on the one hand typographic models to visually highlight the presence of terms and/or definitions, and on the other hand, specific lexical and metalinguistic patterns to connect the elements of the definitional context. This notion has been reinforced in [10], where it is argued that definitional models can in addition, provide certain key clues that help identify the type of definition, which is very useful in developing ontologies.

The **practical studies** implement all these elements in order to develop procedures for the automatic identification and extraction of definitional contexts.

From the point of view of the training mode, we can group the methods in two categories. The first is **supervised approaches**, in which we can place many of the systems based on manually created patterns and rules. As more special approaches, we can mention the method of pattern representation using a model based on word class lattices, presented in [4] where a training set composed of definitions whose formal units are annotated manually is used. Also, in [39] the syntactic dependencies in the sentence are used to create features that describe the semantic annotations of hyponyms and hyperonyms, in order to train a SVM classifier. For the second category, which includes **semi-supervised approaches**, we can mention a number of methods that use the bootstrapping technique such as [14], [23] or [40], all of which, barring some small particularities, apply the general formula of starting from to a small initial set of terms and definitional patterns, iteratively collecting pairs of terms/definitions, from which new patterns are extracted to resume the search cycle.

Depending on the purpose for which they were developed, we also distinguish two categories. The approaches to detecting definitions developed in the context of *question-and-answer* mechanisms [41], [42], [43] are **centered on the *definiendum***, as they look for definitions about a certain term. The second type of approach **focused on the *definitor*** (the verbs that usually appear in definitions), tries to find the complete list of all definitions in a corpus, regardless of the defined terms [5], [13], [44], [40], [3].

The vast majority of approaches to extracting definitions are based on sets of rules or patterns manually created to identify definitions in the text. Besides these, a small percentage of authors sought to incorporate machine learning techniques in an effort to improve the results obtained through the previous application of patterns/rules. Recently, a new research trend has emerged, which completely abandons the use of patterns, relying entirely on the use of neural networks. Next we will review the most important methods, pointing out for each of them their specific characteristics.

For the category of methods in which the **patterns** were **developed and adjusted manually** we mention [26], which describes a definition searching system for the medical field, called DEFINDER, consisting of two modules. The first module, of lexical analysis, applies pattern matching techniques using typical verbal expressions such as "*is called*" and "*is defined as*" as well as a limited set of text markers "()" or "-". The second module, of grammatical analysis, can analyze the syntactic dependencies between words to identify the definitions expressed by more complex linguistic constructions. In [6] the main purpose is to identify the definitional constructions expressed by the hyperonymy relations ("*an X is a Y which* ...") and synonymy ("*equivalent to*"), for which a set of lexical syntactic patterns was created using as information the lemma, the part of speech and the syntactic function of the words. In [45] a web search engine was developed that uses more general and complex models, for example "*cities, such as ProperNoun (Beginning (NounConstruction))*", which make it possible to constrain results by syntactic properties. In [5] the aim is to detect and annotate definitions from a collection of technical German texts. The patterns are centered on definitional verbs, and also specify the positioning of the *definiendum* and *definiens* components, so that, after detecting a definition, they can be easily annotated.

In general, a method based only on rules/patterns has a high accuracy, but a low recall, because it is not able to detect the increased syntactic structural variability that definitions can have. To alleviate this recall problem, [42] proposes the idea of incorporating into the lexical patterns secondary terms that frequently appear in standard definitions, which are collected from dictionaries (WordNet, Enciclopaedia Britannica) as well as from other encyclopedic resources on the Internet. It should be noted that this approach is specific to term-centered methods and is does not scale well to arbitrary terminology and domains.

To avoid the task of manually creating the training dataset, the definitional **patterns** can be **extracted automatically** by various means.

In this direction, a fairly common mechanism for obtaining patterns is the **bootstrapping technique**. In [14] presents an approach involving the automatic estimation (using the bootstrapping method) of patterns in a collection of texts, by generalizing the structure of sentences, using the parts of speech in combination with abstraction masks of non-essential sequences. (". +", ". *"). At each iteration the new patterns are filtered based on a score which takes into account the percentage of initial terms with which they co-appear. At the end, the obtained patterns are subjected to a manual refining process. In [23] a glossary creation system called Glossbot is developed. It uses the web as an information source, being designed to be language independent. To eliminate the need of a domain corpus, the bootstrapping technique is used, initiating the extraction process of the patterns with only a few seed pairs of term - hyperonym. Iterative acquisition of definition extraction patterns from glossary web pages, by exploiting HTML formatting, can cover the great variability of textual definitions, which include both sentences having common lexico-syntactic patterns (e.g. "*a corpus is a collection of documents*"), as well as glossary style definitions (e.g. "*corpus: a collection of documents*"), all these being independent of the target domain. At the end of each iteration, the new definitions are sorted by relevance based on the intersection of the definition word-set with the domain terminology, keeping only those that rank among the first. Also, the bootstrapping technique is used in [40] to facilitate the development of new patterns, but in a hybrid way, being accompanied at the end of each iteration by a manual check of the patterns in order to refine them and, in the last step, to eliminate those with low accuracy. Two types of patterns are used: (1) *lexical* - which extract candidate phrases from the text and *parsing* - used to decide whether the phrase really contains a definition, while also identifying the defined term, its explanation, as well as any other complementary information that is present.

Another possibility to automate the pattern acquisition is to use a corpus previously annotated in a specific way. This is done in [4] which develops a method called word class lattices that allows the extraction of both definitions as well as the hyponym-hyperonym pairs within them. The use of so-called *star patterns* alleviates the variability problem of definitional sentences, while providing a flexible way to automatically extract the hyperonyms from them. To do this, a series of lattice-based classifiers are created, using a set of textual definitions. The training sentences are automatically grouped based on the similarity and, for each such group, a classifier is created in the form of a lattice that models all the variants of the basic pattern of the group. A lattice is a directed acyclic graph, a subclass of automata with non-deterministic finite states. The purpose of the lattice structure is to encode, in a compact form, the important differences between the distinct sequences. Subsequent studies by the same authors have shown that this approach is highly accurate in several areas [46].

As mentioned earlier, a number of methods use **machine learning techniques** to improve the results obtained in the phase of matching patterns/rules. If these results are of low accuracy due to the excessive permissiveness of the patterns, it is possible to use the machine learning module to filter them in order to eliminate false positive cases. In [47] a maximum entropy classifier is used on a corpus of medical pages extracted from the Danish Wikipedia, from which sentences were extracted based on syntactic features. Lexical-syntactic patterns are combined in [48] with a naive Bayes classifier for the purpose of extracting glossaries from tutorial texts in Danish.

In an attempt to solve the problem of language variability, [17] proposes a method based on probabilistic lexico-syntactic patterns, using n-gram models, which can model sequential local dependencies between words. Compared to ordinary patterns, they are capable of generalization and allow for partial matching, by calculating the probability of a so-called "degree of generative match" between a test instance and a set of training instances. In this case, the patterns are not actually used in the definition detection process but only to help the model recognize their specific word sequences.

A very different and unique approach has been proposed by [2] which uses a genetic algorithm to weight a set of definitional characteristics identified by human experts, in order to obtain an efficient definition selection filter.

Remaining in the sphere of machine learning techniques we must mention a recent current of research using only neural networks to model the relationship between term (*definiendum*) and explanation (*definiens*), encoding features through *word embeddings*. This approach is able to improve the recall score, which is quite low for pattern-based methods. The high-level features of the training definitions are encoded via convolutional filters (CNN) and are then provided as input data for the neural network. [49] presents a method of processing lexico-syntactic information between term and explanation through neural networks, in order to learn high-level features capable of identifying similar definitional structures in the text. [3] takes this idea further, creating a complex method of analyzing the similarity both locally between term and explanation and globally between the whole term-explanation and the sentence from which they come.

1.5 Motivations for choosing the technology

In order to create the classifier for the Romanian language, we chose as a source of inspiration the Word Class Lattices method presented in [4], which is based on pattern using. We preferred a pattern-based method in order to be able to access the internal mechanisms of the classifier. Patterns, unlike a machine learning model, can be understood if they are saved in an appropriate format.

An advantage that we considered extremely attractive was the possibility to manually change these patterns after the training phase. In this way, linguistic knowledge can be directly applied to improve the patterns. Also, another positive aspect was the fact that the method is automatically creating the patterns based on a corpus of definitions, thus greatly facilitating the acquisition process. In order to capitalize on this, we also borrowed the corpus of English definitions, which was translated into Romanian, making all the necessary annotation adjustments for this purpose.

1.6 Conclusions

The definition is a description mechanism by which people can understand any unknown term. The essential elements that describe a certain domain can be synthesized in a very brief way through a series of definitions of the main terms of the domain, most often grouped in a glossary.

This chapter reviews the ways in which the definition extraction has been approached over the time, starting with fairly simple manually created patterns, based on the recognition of lexico-syntactic patterns, gradually evolving towards models aiming to automate the creation process of patterns in different ways:

- automatic creation by studying the structure of sentences in a training corpus
- automatic creation using the bootstrapping technique by identifying in text concept – super-concept pairs that are already known

Pattern-based approaches, which to date have proved to be the most accurate, are nevertheless suffering from the disadvantage of having a relatively low recall rate.

To improve this, a current of research has recently begun to unfold, highlighting the modeling power of neural networks specifically designed to process word representations in *word embeddings* format. This format has the ability to encode in a vectorial numerical format the semantic similarities between words.

Being a subproblem of natural language processing, extracting definitions is a very difficult task, which is why, to this day certain types of definitions remain almost impossible to detect. Among them we could mention the type of definitions that extend beyond the limits of a single sentence. The defined term and its definition are in different sentences, being semantically connected by referential expressions. However, in recent years, the line of research based on neural networks seems to be increasingly successful in this direction. This is correlated with the development of a definitional corpus [38] that contains and annotates in a very precise way these difficult cases.

All the approaches presented in this chapter are either exclusively focused on a certain language or have a general character from this point of view. Given this aspect, in this dissertation, we propose as the main objective, the development of a system for definition extraction, specifically optimized for the Romanian language. This system is based on lexico-syntactic patterns created through an automated process using a collection of definitions, manually annotated for this purpose.

2 Contributions to text preprocessing

2.1 Introduction

This dissertation aims to adapt the method of detecting definitions based on word class lattices (WCL), presented in [4], which will be briefly described in the next section.

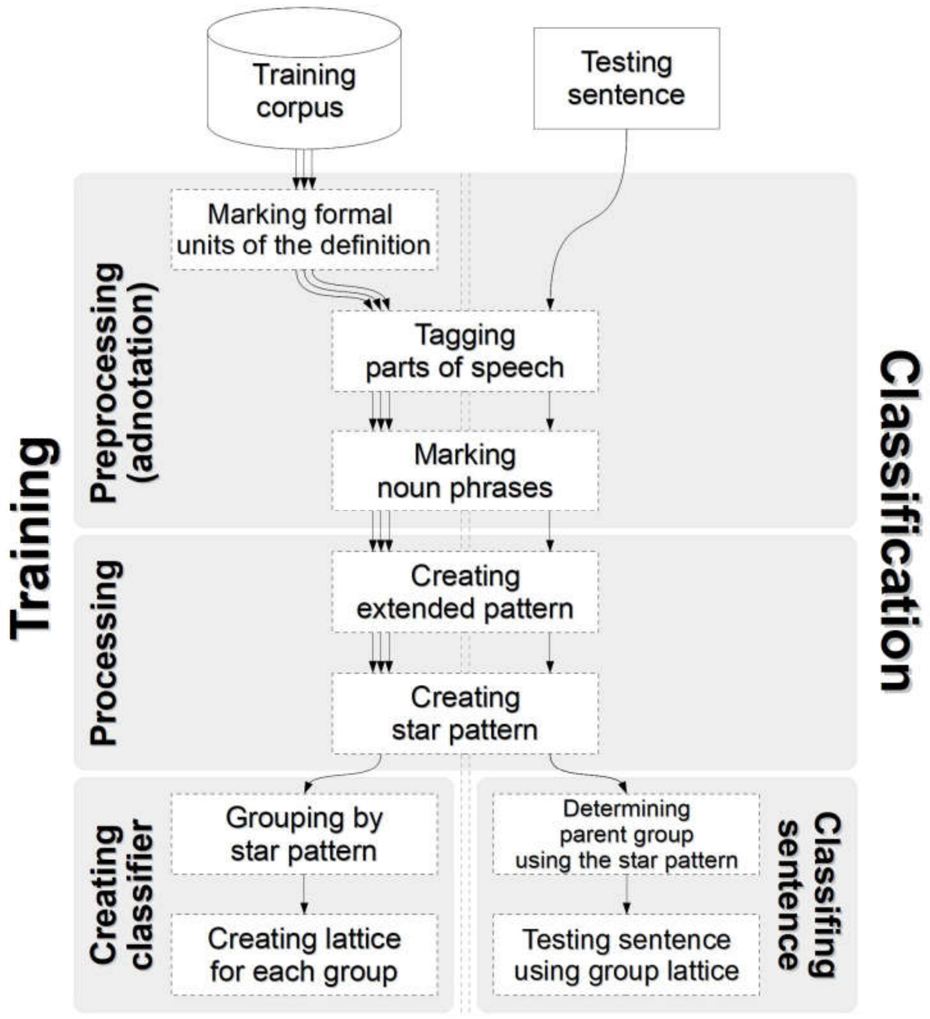
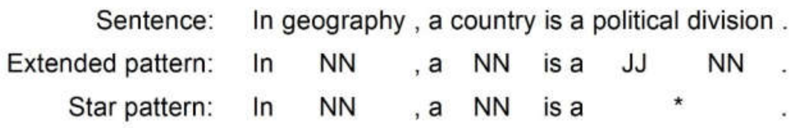

Training corpus
Testing sentence
Training
Preprocessing (adnotation)
Marking formal units of the definition
Tagging parts of speech
Marking noun phrases
Processing
Creating extended pattern
Creating star pattern
Classification
Creating classifier
Grouping by star pattern
Creating lattice for each group
Determining parent group using the star pattern
Testing sentence using group lattice
Classifing sentence

Figure 2-1 WCL classifier diagram

In essence, WCL (figure 2-1) is a pattern-based detection method. The main difficulty in adapting it to the Romanian language stems from the fact that in English the frequency of prepositions within noun phrases is much lower, which allows the creation of so-called star patterns. Creating a star pattern starts with identifying the class of each word in the analyzed sentence. The class of a word is a generalization of that word, which is done following two rules. If that word is very common in the corpus (e.g. "pe / on", "la / at", "de / by"), it is considered a keyword and remains unchanged. Otherwise the word is replaced by its part of speech (e.g. most nouns, adjectives, etc.). The star pattern is obtained by replacing all class sequences of common words with the character “*” (figure 2-2).



Sentence:	In geography , a country is a political division .						
Extended pattern:	In	NN	, a	NN	is a	JJ	NN	.
Star pattern:	In	NN	, a	NN	is a	*	.

Figure 2-2 Star pattern

2.2 Elimination of star patterns

The attempt to create star patterns for the Romanian language has at most a mediocre result, the ability to group the definitions being drastically limited by the frequent use of prepositions. The efficiency of star patterns is given by their power to generalize, which is sufficient for English. By contrast, in Romanian, it is not uncommon for a noun phrase to contain two, three or even more prepositions, which determines causes the star pattern to fragment.

2.3 Reformatting the set of definitions for Romanian language

For the training of the Romanian classifier we decided to use the set of training definitions used for the WCL classifier in English. The initial idea for making this decision was the desire to be able to draw the best possible parallel between the two classifiers. Another reason came from the fact that having the parts of the definition already labeled, they could potentially be used directly after translation without any further manual labeling efforts.

However, this operation encountered two serious problems:

- when translating noun phrases, there were words swapping, prepositions were interspersed, the result being that the formatting for the defined concept and its super-concept was getting in many cases inappropriate, even requiring important adjustments
- a well-established set of annotation rules had not been used for the initial annotation because we found situations in which two structurally similar definitions were annotated differently, which would have led to ambiguities in the training phase.

Considering these aspects, we decided that it was necessary to re-label the definitions taking into account the particularities of the Romanian language, following a well-established set of labeling rules to cover most difficult cases.

The definitions in the training corpus were manually annotated with a series of labels marking off the parts of definition according to the classical model proposed by Aristotle.

The labels used in annotating the formal units of definitions are as follows:

- **<RGET>** - contains the defined word including some additional information
- **<TARGET>** - precisely marks the defined term
- **<VERB>** - specifies the central verbal construction of the definition. This can be just a simple form of the verb "a fi / to be", or it can have a complex structure that contains several (usually two) verbs specific to the definitions.
- **<GENUS>** - includes the part of the text that is the explanation of the defined word
- **<HYPER>** - specifies the word or phrase that designates the category to which the defined entity or aspect belongs
- **<REST>** - designates a part of the text of the definition that contains additional information that can be left out without having an impact on the understanding of the explanation provided by the definition

Differences from the format used for English include:

1. introduction of the pair of tags <TARGET> ... </TARGET> – in English, this group does not exist, using instead a system to replace the defined concept (target) by the keyword TARGET. The change was necessary to support the marking of defined concepts consisting of several words.

2. modifying the <GENUS> tag by adding a corresponding end tag </GENUS> to better process cases where more explanations are provided for the defined word.

3. the group <HYPER> ... </HYPER> does not appear more than once in the same <GENUS> unit anymore. Simple enumerations are grouped into a single <HYPER> unit, and complex cases (in which noun phrases contain prepositions with a low degree of cohesion) are treated using the multiple <GENUS> groups mentioned above.

4. verbal patterns have been simplified as much as possible by introducing special words in the <GENUS> group.

2.4 Considerations regarding the annotation of definitions in Romanian

The main idea behind the re-annotation of definitions is the use of so-called linking words (prepositions, certain adverbs and pronouns) that introduce noun phrases (NP).

Within the definition, these noun phrases represent, by their main noun, the defined concept and it's corresponding hyperonym. In addition, in order to mark off the formal units of the definition, these linking words can be used as helping elements, which in most cases have a fairly well-defined pattern of use.

Usually in the text we are dealing with definitional phrases that contain a definition as well as certain additional explanations, examples, which we can do without. Our goal is to find the minimum number of words that form the main part of a definition, by using the linking words.

In order to explain the procedure for marking off the formal units of the definition (RGET, TARGET, VERB, GENUS, HYPER, REST), we will consider the linking words as decision elements that may decide to remain in a certain formal unit or to transition to the next valid unit. This is dictated by applying the specific rules of the category to which the linking word belongs (e.g. TARGET -> RGET, VERB -> GENUS, HYPER -> GENUS, etc.).

The characteristics of the linking words can be analyzed taking into account two aspects:

- degree of cohesion
- definition completeness

The relationship associated with a linking word has a *degree of cohesion* dictated by the semantic coherence of the respective formal unit and can be classified into two categories:

a. indivisible relations - the unit cannot be divided into subunits without losing its original meaning (e.g. apă ***de*** mare / sea water)
b. divisible relations - can be divided (e.g. formațiuni de rocă ***în*** pământ / rock formations ***in*** the ground)

In terms of *definition completeness*, the aim is to include the minimum amount of information that makes the definition sufficiently informative and intelligible. In this regard, there are three types of information:

a. compulsory
b. informative
c. dispensable

Considering the two aspects described above, (1) the **degree of cohesion** and (2) the **provided information importance**, we can group the linking words into four general categories.

Obs. It should be noted that, in this context, the importance of information is to some extent subjectively assessed, based on the contribution to the overall meaning of the definition, which is being reflected in the manual annotation of the training set of definitions.

The four categories and their default characteristics (rules) are presented below:

STRONG – The information entered by the linking word (e.g. the preposition “de / of”) is mandatory and cannot be separated from the nouns to which it belongs, forming indivisible relationships. For this reason, although they can be found anywhere in the definition, these linking words do not normally lead to a change to another formal unit.

MEDIUM – Also, in this case the information has an important contribution to the degree of understanding of the definition, which makes it belong to the mandatory category. The linking words in this group (e.g. the genitive pronoun “al / of”) are associated with divisible relations. The default rule in their case is that they determine the transition from TARGET to RGET and from HYPER to GENUS, but never from GENUS to REST. Normally, they should not be in TARGET or HYPER.

WEAK – The information entered falls into informative category because, despite helping to add more characteristics to the defined concept, it can be omitted without having a major impact on the overall understanding of the definition. Also this category of linking words (e.g. the prepositions “din / from”, “pentru / for”) corresponds to divisible relations. Just like the MEDIUM category, they determine the transition from TARGET to RGET and from HYPER to GENUS. The difference is that if we are in GENUS, any second linking word encountered will make the transition to REST. By applying the default rules, it follows that these linking words can only appear in RGET, GENUS and REST.

STOP – The information entered does not have a significant impact on the understanding of the definition, being part of the dispensable category. The linking words (e.g. the adverb “cum / how”, the conjunction “iar / and”) correspond to divisible relations and determine the immediate transition to the REST state, regardless of whether we are in HYPER or GENUS.

The linking words, more or less comply with the default transitional rules mentioned for each category. To deal with the rather numerous exceptions, specific rules have been developed for each linking word.

2.5 Contributions on describing the use patterns of linking words

Using the following three statistics: the number of occurrences (*Nr_total*), the number of transitions (*Nr_trans*) and the number of exceptions (*Nr_exc*), three measures were defined to describe the linking word patterns of use:

1. ***Fitness*** - the matching degree of linking words to their category

The matching degree measures the degree to which the linking word complies with the rules set for the category to which it belongs. It is measured in percentages, the calculation formula being as follows:

$$Fitness = \left(1 - \frac{Nr\_exc}{Nr\_total}\right) \cdot 100$$

Using this measure, it is possible to determine how appropriate was to classify a linking word in a certain category and possibly decide on its re-classification in a more appropriate category.

From a probabilistic perspective this measure can be interpreted as the predictive power of that linking word regarding the default annotation rules that should be applied.

2. ***Mobility*** – the mobility of linking words

Mobility is a measure of a linking word predisposition to determine the transition from one formal unit to another within the definition. It is calculated by making the ratio between the number of transitions and the total number of occurrences as follows:

$$Mobility = \frac{Nr\_trans}{Nr\_total}$$

As can be seen from the formula, its values vary in the [0, 1] range, where 0 indicates immobility and 1 is total mobility.

3. ***PredError*** - transition prediction error for linking words

The prediction error is the proportion in which the linking words appear in situations that violate the default rules of the category to which they belong, which causes parent definitions become more specific or general (depending on the type of exception in question).

The default rules, which are quite general and have the role of global directives, cannot cover all the particularities of each linking word. For this reason, the exclusive application of these rules, which are intentionally simple, tends to result in the extraction of vague definitions. On the other hand, the annotations made in the training set take into account the individualities of the linking words, thus creating the differences measured by the prediction error, highlighting the linking words that have special characteristics.

From an informational point of view, linking word deviations can have two tendencies, with respect to the inclusion of information, as prescribed by the default rules:

- more – thus increasing the specificity of the definition
- less – making the definition more general

In practice, two values are calculated for the prediction error, corresponding to the trends described above:

$$PredError_{+} = \frac{Nr\_exc_{+}}{Nr\_total}$$

$$PredError_{-} = \frac{Nr\_exc_{-}}{Nr\_total}$$

The values for Nr_exc_{+} și Nr_exc_{-} are computed using the following two formulas:

$$Nr\_exc_{+} = Nr\_exc_{TARGET\_RGET} + Nr\_exc_{HYPER\_GENUS} + Nr\_exc_{GENUS\_REST}$$

$$Nr\_exc_{-} = Nr\_exc_{RGET\_TARGET} + Nr\_exc_{GENUS\_HYPER} + Nr\_exc_{REST\_GENUS}$$

Calculating the prediction error for a series of common linking words shows the almost exclusive tendency to include information, which is perfectly natural, considering the fact that a definition must be as precise (specific) as possible in order to effectively provide the intended explanation.

Because the values for $PredError_{+}$ și $PredError_{-}$ are normalized, we can define the degree of trust (TrustDegree) as the inverse measure of the total prediction error:

$$TrustDegree = 1 - (PredError_{+} + PredError_{-})$$

2.6 Conclusions

Contribution list for chapter 2:

- adapting the way of annotating the formal units of the definitions for the Romanian language
- identification of linking words and grouping them into four categories: STRONG, MEDIUM, WEAK, STOP
- elaboration of measures to describe the usage patterns of linking words
- elaboration of a comprehensive set of rules for annotating the formal units of definitions, taking into account the important particular characteristics of the linking words

This chapter describes the adaptations made to the method of extracting definitions, presented by Navigli and Velardi in [4], which is based on patterns grouped using word class lattices. Extracted patterns are called *star patterns* due to the fact that they replace certain part of speech sequences (to increase the generality) by the character "*".

For the Romanian language, the use of the method based on star patterns is inefficient due to the increased frequency of prepositions. To obtain an efficient alternative type of pattern, we introduce the concept of *linking word*. Four categories of linking words are identified according to the associated *cohesion degree* (indivisible and divisible relations) and to the *information importance* introduced by them (mandatory, informative, dispensable). These categories were named STRONG, MEDIUM, WEAK and STOP, each having a specific combination of values corresponding to the two traits mentioned above.

The adaptation of the method presented by Navigli and Velardi involves some changes that are essentially related to the way in which the defined concepts and super-concepts are marked.

These differences, as well as the careful study of the English corpus (which was translated in order to be used for the adapted Romanian method) gave rise to the need to develop a set of rules for annotating the formal units of the definition, to ensure the extraction of patterns having a consistent format.

Also, in order to describe in more detail the linking words, a series of measures were developed, namely: *fitness* (matching degree), *mobility*, *predError* (prediction error) calculated based on statistical numerical data extracted from the training corpus, such as: number of occurrences, number of transitions, number of exceptions. All these values are calculated for each linking word present in the corpus.

3 Contributions regarding morpho-syntactic labeling tools in Romanian

One of the first stages of natural language processing is the labeling of speech parts (POS tagging). Most approaches are generally focused on sequential word processing. This action involves associating a part of speech (word class, lexical category) with each word. The process can be seen as a simplified form of morphological analysis.

From a linguistic point of view, specialists largely agree that there are three main parts of speech: noun, verb and adjective [52]. When creating a set of labels, the basic idea is to label with speech parts all classes of words that have different grammatical behavior.

Almost all speech part labeling systems work with a well-defined set of labels. Given a sentence, the system must assign a label to each of its words. This action encounters two major difficulties:

- ambiguous words - words that can be assigned multiple tags
- unknown words - words that do not appear in the training corpus

Another problem in labeling the parts of speech is the concordance of the label set. The use of a large set allows for the encoding of several aspects related to the morpho-syntactic structure of words, but at the same time makes it difficult to distinguish between similar labels. This distinction can sometimes be so subtle that even people may disagree in this regard as shown by an experiment done in [53] on Penn Treebank in which the experts who annotated had different opinions in about 7.2% of cases.

The maximum accuracy obtained in labeling (the number of words correctly labeled in relation to the total number of words) is currently around 96% -97% for most Indo-European languages. It is worth noting that it is possible to achieve a high accuracy (around 90%) using only extremely simplistic methods, such as only choosing the most likely word tags [54]. Sophisticated methods aim to cover the last 10%.The accuracy of 96%-97% can still be seen as very high and any attempt to improve performance above these values proves extremely difficult.

In [55], it is made a classification of the various error types preventing the labeling tools from achieving full accuracy (table 3-1), which also include human errors present in training sets:

Table 3-1 Types of speech labeling errors

Nr. crt.	Class	Frequency
1	Non-existent form in the lexicon	4.5%
2	Unknown word	4.5%
3	Possibility of correct labeling	16%
4	Difficult linguistic context	19.5%
5	Imprecision/vagueness	12%
6	Inconsistency/missing standard	28%
7	Wrong gold standard	15.5%

3.1 Analysis of existing tools

All the processing systems analyzed are based in one form or another on the morpho-syntactic description tag set (MSD) [56] developed within the Multex-East project (www.nl.ijs.si / ME).

For the operation of labeling and chunking of speech parts we evaluated the following three tools:

1. **UAIC POS tagger** [57] is a hybrid labeling tool that combines a statistical model with a rule-based system, which is the main element of this approach. The rules are being used to reduce the ambiguity of words before the final labeling operation. This greatly facilitates the action of correcting certain types of errors, as it allows the direct input of linguistic skills into the system. In addition to the tagging tool, a tool for chunking the noun phrases has also been developed [58].

2. **MLPLA - Modular Language Processing framework for Lightweight Applications** [59] is a modular processing tool. It contains an input module, a series of processing modules (part of speech tagger, lemmatizer, chunker, etc.) whose operation is independent of each other, offering the possibility of creating flexible processing lines, depending on the needs. Finally it contains an output module for information formatting. MLPLA uses a small set of MSD-derived tags, called CTAG [60] in which certain information such as gender is excluded from tags because it can be inferred as needed based on the word form. The number of labels is thus reduced to 78, but this increases the degree of ambiguity in labeling.

The new tools developed within RACAI, MLPLA v2 and RELATE, are trained using the methodology and annotations specified by the Universal Dependencies project (universaldependencies.org). For this reason, many of the labels used by these tools correspond to those in English. Unfortunately, at the time of writing, they are only accessible as a web service, which is a major impediment to the processing needs of the method developed in this dissertation.

3. **TreeTagger** [30] was designed in the early ‘90s in response to the problem that almost all previous labeling tools had, namely the accurate estimation of low-value probabilities, given a small training set. The proposed technique to avoid the problem of data scarcity is based on a decision tree to accomplish a highly reliable estimation of transition probabilities.

Although it was not developed specifically for Romanian language, it can be trained like any other tool of this type for the language of interest and can even obtain good results, if the training corpus has the appropriate in size relative to the set of target labels.

TreeTagger does not offer facilities for chunking.

4. **UDPipe** [60] is a trainable tool for tokenization, labeling, lemmatization and text dependency parsing of texts in CoNLL-U format. The main objective in its development was to make it as easy as possible to use, which is why the following aspects were taken into account:

- provide the best tools for tokenization, morphological analysis, part of speech labeling and dependency parsing
- create a single tool using a single model (for each language)
- create models for as many languages as possible
- avoid using language-specific information

UDPipe accuracy for the Romanian language (version 2.0) is 81% for the parts of speech labeling and 75% for lemmatization.

In order to verify the text labeling tools, a series of definitions from the English WCL training corpus were chosen at random. In order to exemplify the type of information and format provided as the output of each tool, we have chosen the following definition:

In Greek mythology, Callisto was a nymph of Artemis.

The definition was first translated into Romanian and then processed with each of the labeling tools mentioned above in order to compare the results and determine the reliability of each:

UAIC POS tagger

The hybrid labeling tool developed at UAIC provides the most comprehensive set of information among all three analyzed systems. In terms of labeling capability did very well, as it registered no errors for the selected definition. The result obtained in the labeling phase of the speech parts can be used directly without format change for the chunking operation.

MLPLA și MLPLA v2 / RELATE

The tool handles the tagging job surprisingly well, taking into account that the main restrictive parameter in its development, was the reduction as much as possible of the model size and of the computational costs. In the case of the analyzed definition, this tool registered one labeling error.

The new version of this tool, MLPLA v2, as well as RELATE, trained using Universal Dependencies have similar results to UAIC POS tagger.

Also, the use of a label set similar to the standard used for English gives it an advantage over the other tools, because in this approach we want to keep a high degree of parallelism between the original English method and its adaptation for Romanian. In addition it provides a wider range of information that is extremely useful in processing definitional sentences.

TreeTagger

The information provided by TreeTagger is minimal, including only the MSD label and word tag. It also registered two labeling errors.

Another drawback was that being a standalone application, it needed launching an external process to function, the transfer of input / output information requiring an indirect and cumbersome interaction through files on hard-drive.

UDPipe

UDPipe also provides a generous set of morpho-syntactic information. Like MLPLA v2 / RELATE it was also trained using Universal Dependencies, providing both MSD and UD tags. Just like them, it can also analyze the syntactic dependencies between the sentence terms.

In order of importance, we will list the priorities for the method we want to develop:

1. the possibility of integrating the labeling tool into a stand-alone application
2. performance in labeling the parts of speech

Regarding the first aspect, we must specify that MLPLA v2 / RELATE are available only as web services and as such they do not allow their integration into an application. TreeTagger, being an independent application itself, requires the launch of an external process to operate, the transfer of input/output information requiring an indirect and cumbersome interaction through files on disk. Although UDPipe is written in C++, it provides a binding system to allow access from Java. Finally, the MLPLA and UAIC POS tagger come in the form of Java libraries that can be easily integrated into any project, especially the latter that can be used without any prior adjustment.

In terms of labeling performance, MLPLA v2, RELATE and UDPipe can be considered on the same level, closely followed by UAIC POS tagger. Given that MLPLA v2 and RELATE have been removed from the race, the final comparison is between UDPipe and UAIC POS tagger.

Normally, the winner would have been UDPipe due to its good performance and relatively simple method of integration. The reason why it was not actually used in the thesis is that this tool was discovered some time later in the development process, at which time important efforts have already been made for integrating of the previously chosen tool, **UAIC POS tagger**. In addition, although the test went well, the success rates for labeling the parts of speech and lemmatization, as presented on the site, appeared to be lower than those corresponding to the UAIC POS tagger. For these reasons, we considered that the time that should have been allocated to the reintegration of a new labeling tool should be used for activities closer to the research goal.

3.2 Improving the accuracy of POS tagging task for Romanian language

The ease of integrating the chosen labeling tool, UAIC POS tagger, into a Java application isn't by far its most important feature. As previously mentioned, information processing is performed in two stages:

- a statistical step - performed with the help of a maximum entropy classifier
- a rule-based step that processes previous results

This second step is an extremely important element, as the rules can be created and adjusted manually to increase the labeling accuracy.

The rules are in fact a series of complex patterns, represented in the form of directed graphs, each pattern component being located in a graph node. Some nodes may decide to choose desired variants of speech parts (KEEP), others to eliminate them (REMOVE) as needed. The graph is traversed from right to left, and where there are branching paths, they are pursued based on a ordering number associated with the link. The search fails if no path is found to traverse the graph from beginning to end.

Although the implicit rules of the tagging tool were normally appropriate, we noticed that while processing of the definitional sentences, some common errors were routinely cropping up during both tagging and chunking phases, which made us start a process of adding and amending rules to alleviate these shortcomings.

Obs. Further on, the references to the *direct* forms mean the cases of nominative / accusative, and the *oblique* ones, the cases of genitive / dative.

Rules for correcting the POS tags

The adjustments made to the rules of parts of speech identification determine the change of analysis output of a number of 1084 definitions out of the total of 1773.

We cannot say whether all these really constitute positive changes, because, no matter how carefully the new rule is analyzed, negative side effects could occur. In the vast majority of analyzed cases, the results obtained by implementing the new rules were certainly positive.

We will mention some of the changes with an impact that is worth mentioning:

- rule for correcting the error by which the conjunction "*sau / or*" was almost always confused with the non-diacritic form of the "*său / its*" determinant, which was extremely serious given that the conjunction *or* plays a fundamental role in identifying the enumerations - 501 errors
- rule for the correct identification of infinitive verbal forms that were assigned the label for the indicative form - 172 errors
- rule for dealing with the error by which the genitival article "*al*" was labeled as the demonstrative article "*cel*", being confused with the non-diacritic form of regionalism "*ăl*", a very serious aspect because the genitival article is considered a MEDIUM type linking word - 115 errors
- rule for improving the error by which words that do not exist in the tagger's dictionary

were labeled as adverbs, which is corrected by considering that such words located in the first position in a sentence are usually proper nouns - ~ 90 errors

- rule for correcting the classification of the word "*multe / many*" in the incorrect oblique category, which determines in turn an erroneous marking of noun phrases - 33 errors
- rule for correcting the error by which the adjective "*mari / big*" was misidentified as the form without diacritics of the noun "*mări / seas*" - 24 errors
- rule for correcting the classification of nouns with direct form preceded by oblique determinant in the category of oblique nouns - 11 errors

Adjustments to noun phrase chunking

Changes in this category have been channeled into improving existing rules in order to accept new possible combinations of terms within noun phrases. In the end, all these adjustments resulted in marking correction of approximately 560 noun phrases.

Since we cannot talk about distinct rules, we will describe the successive steps leding to the final result:

- inclusion of adverbs located after adjectives and the possibility of having several determinants before a noun - 48 corrections
- inclusion of additional situations related to the degrees of comparison before nouns and adjectives with direct form associated with oblique nouns - 222 corrections
- inclusion of indefinite articles separated from the noun by adjectives or determinants, and of ordinal numerals located before nouns - 164 corrections
- inclusion of determinants located after oblique nouns, improving the integration of nouns accompanied by the genitival article "*al / of*" - 86 corrections
- inclusion of nouns that are part of an enumeration but lack the genitive article - 35 corrections
- improving the integration between common and proper nouns - 40 corrections

3.3 Conclusions

Contribution list for chapter 3:

- improving the rules for correcting the labels corresponding to the parts of speech
- improving the chunking of noun phrases

This chapter discusses the characteristics of labeling tools as well as the causes of errors in the process of labeling the parts of speech.

With this in mind, we perform an analysis of the features and performance of five labeling tools: UAIC POS Tagger, MLPLA - Modular Language Processing framework for Lightweight Applications, MLPLA v2 / RELATE, UDPipe and TreeTagger, to determine which is the most suitable and best meets the needs of our application for definition detection for the Romanian language, developed in this dissertation.

Following the tests, UAIC POS Tagger is chosen due to its flexibility and ease of integration into a standalone application. This labeling tool has the particularity of having a hybrid nature, its operation being carried out in two stages: a statistical stage based on a linear regression classifier trained using the maximum entropy principle and a rule-based stage that allows the adjustment of end results. This way the user can greatly influence the results of the first stage by adjusting these rules.

As contributions in this direction, we have created a series of new rules resulting in both the improvement of the accuracy in the parts of speech tagging operation as well as the increase in efficiency of chunking the noun phrases.

4 Contributions on the representation language of simplification patterns

4.1 Introduction

The method presented in this dissertation is based on a test system using *definitional patterns*. They are created using a set of *simplification patterns*, based on information extracted from a collection of texts consisting exclusively of definitions. The process of lexical simplification of sentences is performed in order to bring them to a *canonical form*, so that the resulting definitional patterns are as general and as simple as possible, while retaining the essential characteristics of a definition.

In the previous paragraph we mentioned two types of patterns:

3. **simplification patterns** - used in the training phase to obtain the canonical form of sentences
4. *simple definitional patterns* (**simple patterns**) - created in the training phase based on the canonical form and which will be used in the classification phase for the detection / extraction of definitions from the text

These simplification patterns can be seen as simplification rules that will be applied sequentially to obtain the canonical form of the sentences. They are specified in external configuration files to facilitate their editing. A relatively simple representation language was specifically designed to define the rules, with a compact syntax centered on the pattern component. We can say that the most important characteristic of language is *flexibility*, which is extremely useful in the phase of elaboration (experimentation) of simplification patterns.

In order to control the simplification process, a word annotation mechanism is provided through which information can be attached in a coded way for the rules that will be used further on. Unlike the simplification rules that are associated with the whole pattern, the annotation rules are associated only with a certain component of the pattern.

The classifier described here is using three such simplification pattern configuration files:

- for representing patterns associated with linking words (LINKWORDS)
- for representing the patterns for simplification and annotation rules (MIXED)
- for defining complex pattern components (SETS)

Given that they perform specific functions, each of them has certain syntactic features, which is very obvious in case of the SETS file.

Each file consists of two main sections:

1. the first of them defines directives for preprocessing rules
2. the second contains the actual patterns (LINKWORDS, MIXED) or the set definitions (SETS)

In this chapter we will discuss in detail the simplification patterns and we will make numerous references to their components to explain their types and mode of action.

For this reason, in order to ensure the clarity of the text as much as possible, we will make the following symbolic notations for the simplification pattern and its components:

- ***Spattern*** - simplification pattern
- ***sTcomponent*** - simplification type component

4.2 General aspects related to the syntax of pattern representation language

In configuration files, sections are specified by keywords preceded by the @ character. They are structured, as mentioned earlier, into two main sections, which in turn contain a number of subsections.

The following two directives correspond to the two main sections:

1. *@preprocess* - for text preprocessing
2. *@structure* - for the rules part

The complete syntax of each also contains the name of the root node (e.g. *@structure: MIXED*)

The appearance order of the two sections in the file has to be the one mentioned above.

The end of any section can be marked with the *@end* keyword, but this is not required. Its use is only necessary when defining a series of nested sections.

Comments can appear anywhere in the file:

- single line, preceded by the sequence “//”
- of several lines, the beginning being marked by “/**”, and the end by “**/”

4.3 Contributions regarding the information representation in text preprocessing task

The preprocessing rules are grouped in a subordinate section, in the preprocessing part, called *@adjustments*. In the current version, only one such rule has been added, helping with the processing of words containing diacritics.

Also, as a preprocessing method, it is possible to replace certain text sequences with others. These rules are grouped in the *@replacement* section, each written on a single line containing the text to be replaced and the replacement text. They are used to generate complicated text sequences in the file that can appear in several places, thus helping improve the readability of the text.

These preprocessing rules operate at string level.

1. **LINKWORDS** - representation of patterns associated with linking words

As we presented in the previous chapters, the linking words are the main connecting elements in the sentence. Given their importance, their associated patterns were defined in a separate file in order to better meet their preprocessing needs.

Although there are no restrictions on the full use of language representation facilities, the patterns are simple, as the linking words consist of one, rarely several words. This is largely due to the fact that the part of speech labeling tool is grouping the words of complex prepositions using the character "~", thus creating a single compound word (e.g. "*astfel~încât* / so~that").

Linking word patterns are specified in the *@structure: LINKWORDS* section and are grouped by the four main categories described above: STRONG, MEDIUM, WEAK and STOP, using the *@type* keyword. For example for the STRONG category we have the following sequence:

```
@type:STRONG
  ...
@end
```

Language allows the discontinuous definition of a category if so desired.

The definition of lining word patterns and their classification into categories is based on the observations gained from the analysis of definitions in the training corpus. Most of the linking words are prepositions and the most common ones were obtained from the training corpus. In order to have a complete list, the rest of the prepositions were extracted from the dictionary used by the tagger (which exists in the form of a text file), using REGEX queries to isolate the words of interest.

2. **MIXED** – representation of simplification and annotation rules

As in the previous case, the file for simplification rule definition has in the main structure section *@structure* a series of subsections in which the rules are grouped according to the purpose of use or the moment they are used in the process of creating the canonical form of the processed sentences.

The syntax is the same as for defining the linking word categories. Such a subsection is marked with the same keyword *@type* followed by its name (e.g. *@type: REMOVABLES*) which identifies the role played in the sentence transformation process.

In addition to the simplification rules, there are also helpful rules, called annotation rules. They do not directly simplify the sentence, but only label certain words or word constructions so that they can be later identified and specifically processed (ex. 4.1). In general, a simplification pattern will contain besides the specific simplification rules, some annotation rules, the two types being often inseparable.

Ex. 4.1 l:egal->OVRD ?= cu

The modality of using the defined patterns takes into account the order in which they appear in the configuration file. Obtaining the canonical form is a sequential process and in general the successful application of some rules depends on certain previous steps.

3. **SETS** – defining complex pattern components

In addition to the main files of pattern configuration, there is a third secondary file that allows the defining of complex pattern components (sets of simple pattern components).

It should be noted that the pattern definition language also provides a localized way to create such sets, but they are anonymous and valid only within the pattern to which they belong.

On the other hand, the way to define sets in the SETS file, although more complex, has the following notable advantages:

- associates a name for each set so that it can be used as many times as necessary, thus eliminating the restriction of local use
- in addition to the usual elements that are part of anonymous sets, namely: words, lemmas, and parts of speech, a set can also include other sets defined in the set file.

As a relative disadvantage we can consider that the definition in a separate file makes it difficult to immediately understand the pattern in which such a set is used.

In terms of syntax, the set definition file has many similarities to the two pattern files, having:

- a part for preprocessing that is identical
- a part for defining the sets that is structurally similar with those of pattern files

4.4 Simplification/annotation patterns

Patterns are made up of a series of space-separated pattern components.

There are two categories of *sPcomponents*:

1. of control - introduce various information that influences the interpretation of patterns
2. of structure - are the elements that will be actually used in the process of testing the word sequence

From a structural point of view, the patterns consist of three parts:

- precondition
- actual pattern
- post-condition

Among them, only the actual pattern is required to be present, the precondition and post-condition are optional. Although these three parts are identical in terms of syntax, they differ in their interpretation, which is related to the interaction with the words in the analyzed sentence, during the alignment process.

Usually the patterns are short enough to fit on a single line, but the language provides a mechanism to write the pattern on several lines using a special operator.

4.5 The syntax of pattern representation language

Control *sTcomponents*

They control how the pattern works as a whole, so we could consider them meta-elements.

Table 4-1 summarizes these control components:

Table 4-1 Control components

Pattern component	**Description**
?=	precedes the precondition or post-condition
=	precedes the main part of the pattern
#< name>	associates a name to the pattern
@< directive>	associates a specific directive to the pattern

The *sTcomponent* ”=” is required to appear only in cases where there is a transition between two parts of the pattern (e.g. the pattern consists of precondition and actual pattern), otherwise it can be left out. On the other hand, in the current implementation, “? =” must always be present to mark the beginning of the precondition and post-condition.

If we want to use the same pattern in multiple times within various other patterns, without having to rewrite it each time, it is necessary to name it via the control component "# <*name*>". This component must be in the first position in the pattern.

The control *sTcomponent* preceded by "@" also appears in the first position. It’s status is special, because it has a polymorphic nature being able to introduce a series of directives that fulfill various extremely different functions:

- @debug <n> - where <n> is a natural number marking the pattern to make it’s identification possible when activated
- @repetable – modifies the pattern application routine by trying to reactivate it as many times as possible before moving on to the next pattern

Structural *sTcomponents*

There are three types of structural *sTcomponents*:

1. inner-pattern type - consists only of a name reference to a pre-defined pattern, which was marked with a name for later easier reference
2. set type - groups of simple *sTcomponents*
3. simple *sTcomponents* - the basic components of the *Spatterns*

A simple *sTcomponent* contains the core, a series of modifiers, and annotations.

The **core** is the main part of a simple structural *sTcomponent*. In turn, it contains the following elements:

- a prefix that specifies the type of the structural *sTcomponent* - can be "w", "l" or "p"
- the word, the lemma, or, respectively, the part of speech in question
- an MSD label that can establish the accuracy of the testing

The character ":" is used to connect and at the same time make the distinction between the prefix and the word, the lemma or the part of speech. (e.g. p:ADVER)

Using the prefix "w" for the word is optional because, it is implicitly considered that *sTcomponent* of the word type, if there is no prefix.

The part of speech is specified using only the first 5 characters of its name. If the name is shorter, it remains unchanged.

The syntax for integrating the MSD tag into the component uses the "/" character that frames the label without spaces before and after. While testing, the matching of MSD labels is performed using REGEX expressions.

In case we are not interested in the core type, we can use the unknown type. In the current implementation this type is only allowed in combination with an annotation test in order to verify the existence of a certain label.

Modifiers are operators that control how the *sTcomponent* is interpreted.

There are two categories of modifiers that can also be identified depending on their placement relative to the component core:

- of control - located before the main part
- operative - placed after the main part

Table 4-2 lists the available modifiers.

Table 4-2 Modifiers of sTcomponents

Category	Function	Symbol
of control	target *sTcomponent*	>>
	excluded *sTcomponent*	!
operative	recurrent *sTcomponent*	+
	optional *sTcomponent*	?

The **target** *sTcomponent* modifier marks the main component of that pattern. It can only be used in the main part of the pattern. It can have two uses:

- in case of patterns that simplify the phrase, this component marks the main word used as a substitute
- when using the annotation with default tags, it marks the word to be annotated

Obs.
1. There may be only one target component in a pattern
2. It is perfectly possible for a pattern to apply both the simplification and the annotation operation to the target word.

The **excluded** *sTcomponent* modifier marks the components that will be ignored in the simplification operation described in a pattern.

The **recurring** *sTcomponent* modifier is an element that allows matching with multiple words of the same type.

The **optional** *sTcomponent* modifier determines that the respective component is allowed to be skipped for testing.

Annotations provide a way to store information resulting from the application of a pattern.

The use of annotations involves two types of operations:

- defining - a certain label is assigned to a word
- testing - checking if a particular word has a specific label

Table 4-3 shows the operators involved in the annotation process.

Table 4-3 Annotation operators

Operation type	Specific function	Symbol
defining	additive	->
	overwriting	->*
	defining pattern element	->:
testing	positive	#
	negative	#!

Defining operators can associate, through annotation, certain symbolic information to the words in the simplified phrase. A word can have an unlimited number of annotations. The additive operation type is the main way to add the annotations.

By contrast, the overwriting operator deletes all previous annotations, replacing them with the new one provided as a parameter.

In addition to the usual annotation operation, it is possible to force the setting of the string for the label of the simple pattern component, using the last defining operator, thus canceling the usual procedure for obtaining it.

Test operators verify the presence (in the case of a positive test operator) or the absence (in the case of a negative test operator) of a certain annotation.

Obs. The syntax allows the simultaneous definition and testing of several annotations, using the pipe operator "|". In the case of testing, each label tested may be negative or positive as needed. In this sense, the operator "!" will be associated with each annotation when a negative test is required:

Ex. 4.2 p:NOUN#ART_U|!OBLQ

4.6 Conclusions

Contribution list for chapter 4:

- creating a language for representing sentence simplification patterns

In this chapter we present a special language created to represent the patterns that make possible the sentence simplification. This is done in order to obtain a canonical form, that will allow the creation of patterns with a lower degree of complexity.

This language defines a series of term-centered syntactic elements that specify the way of interaction between pattern components and how these components match the terms in the analyzed sentence.

The sentence simplification rules are specified in three configuration files, each with its own characteristics. These files are called LINKWORDS, MIXED and SETS.

The LINKWORDS file contains the patterns associated with the linking words. They are grouped into sections according to the linking word type.

The MIXED file contains the representation of the patterns for the simplification and annotation rules. These are rules of high complexity and are grouped according to the step for which they were designed. There is generally a well-established order between these steps as certain processing outcomes can and will generally be used in later phases. Unlike the order of the steps that is dictated programmatically within the application, the order of the rules within a step is determined based on the order of occurrence in the configuration file.

In the SETS file, complex pattern components that can be used within the patterns can be defined. These are complex data collections that can contain words, parts of speech, as well as other sets.

The syntax of the pattern representation language defines two types of components: *of control* - which controls how the pattern works; and *of structure* - which are the actual elements of testing.

The structural components have a core that determines the type of test, as well as a series of modifiers that determine how this testing will be performed. Each structural component can perform annotation operations on the corresponding sentence term or can check if the term possesses a certain annotation.

5 Contributions regarding the process of obtaining simple patterns

5.1 Introduction

The main steps of the Romanian language method for definition detection are largely the same as those of the WCL method whose diagram was presented in chapter three (figure 2-1). Notable differences are registered in only two of these stages.

As we mentioned before, the first and by far the most important difference is that instead of the star pattern, an equivalent pattern called the *simple pattern* is created. In figure 5-1 we present the sub-stages of creating the simple pattern, which is an important contribution of this dissertation.

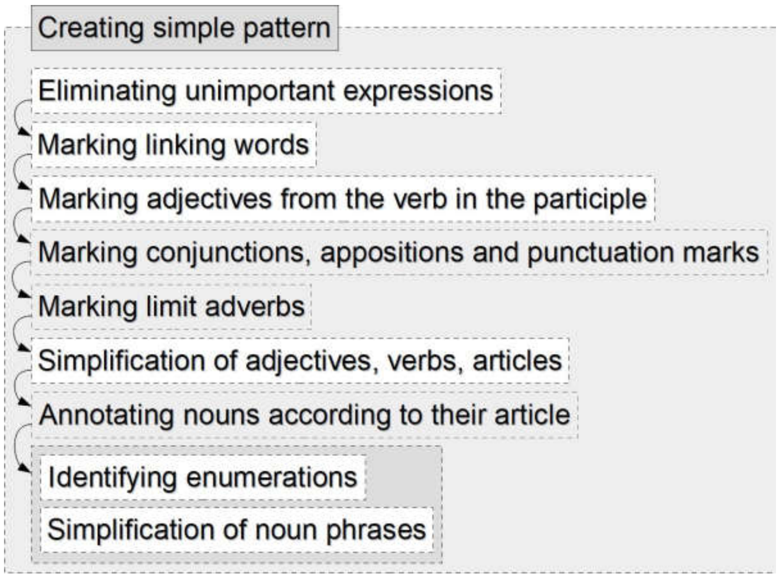

Creating simple pattern
Eliminating unimportant expressions
Marking linking words
Marking adjectives from the verb in the participle
Marking conjunctions, appositions and punctuation marks
Marking limit adverbs
Simplification of adjectives, verbs, articles
Annotating nouns according to their article
Identifying enumerations
Simplification of noun phrases

Figure 5-1 The steps for creating the simple pattern

The second important difference is in the preprocessing phase, in the part of speech labeling step. Before sending the sentence to the labeling tool for analysis, we perform specific adjustment operations on certain words, word sequences or punctuation marks to help correctly identify the corresponding parts of speech (figure 5-2).

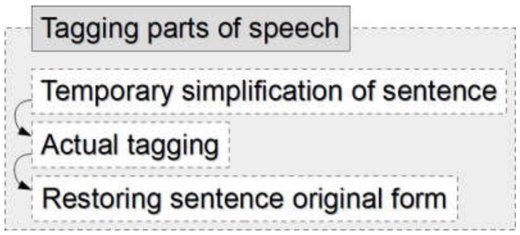

Tagging parts of speech
Temporary simplification of sentence
Actual tagging
Restoring sentence original form

Figure 5-2 Special preprocessing operations

In this chapter we will talk a lot about two types of patterns. The first is the *simplification pattern* that we discussed in chapter 4, and the second is the *processing pattern* which is the current iteration of the pattern for the analyzed sentence. It starts from the initial form, called *extended pattern* and to the final form of *simple pattern*.

Similar to the notations we made in chapter 4, we will use abbreviated, symbolic forms to refer to the pattern and its components:

- ***Ppattern*** - processing pattern
- ***pTcomponent*** - processing type component

We will also use the notation ***Tcomponent*** to refer to a generic pattern component.

5.2 Internal representation of the *Ppattern* - structure, operations

To create the sentence pattern, we need a structure that allows simplification operations, while being able to "memorize" all the successive stages of transformation, from the extended pattern to the simple pattern form. This structure has been specially designed for this purpose. It is implemented by means of a lattice of *Tcomponents*, which initially correspond to the terms in the analyzed phrase (figure 5-3). This lattice is an acyclic directed graph with a single starting point. It contains two types of nodes:

- data nodes - contain the *Tcomponents*
- connecting nodes

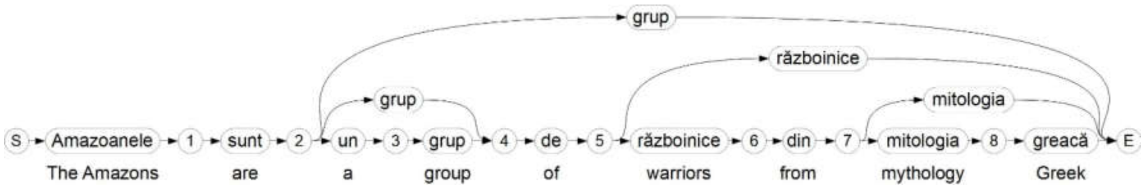


Figure 5-3 TComponent lattice

Each connecting node has at least one input and one output connection. When a simplification operation is performed, a new node is created that inherits the data of the target *pTcomponent*. It is added between the two link nodes at the extremities of the simplified sequence.

The current pattern (*patternP*) always contains the components on the upper levels.

The internal representation supports the structure traversal both from left to right and vice versa, which makes it possible to perform operations that would otherwise be difficult, if not impossible.

In order to carry out the operations necessary to obtain the pattern of canonical form, a series of simple operations for transforming the lattice structure were defined. They mainly work on data nodes, and the link nodes, which are transparent in terms of lexico-sintactic information, undergo related collateral changes.

Given that these operations change the structure of *pTcomponents* there is a restriction that prevents any kind of operation on components that are not at the highest level.

The defined atomic operations are as follows:

1. Simplification of a sequence of nodes
2. Removing nodes
3. Adding nodes
4. Interchange of nodes
5. Repositioning a node before / after a target node

5.3 Activating the rules of pattern simplification

Pattern testing is the operation by which the patterns defined in the rule configuration files (primarily MIXED, using the sets defined in SETS, but also LINKWORDS) are successively activated and tested to see if they match a certain sequence of words in the sentence.

In the process of comparing the *Spattern* with the *Ppattern*, the test operation is successfully completed when all the partial tests specified by the components of the *Spattern* are fulfilled. Because there are optional and/or repetitive pattern elements, it is not possible to determine a priori the number of *pTcomponents* that will be affected in the end.

Whatever the structure of a pattern, there are four ways to start the testing. These correspond to the position of the search anchor point, which determines the first testing *sTcomponent*. This element corresponds to the current working *pTcomponent*.

The three structural parts (precondition, actual pattern, post-condition) are tested separately, this being done according to the chosen type of test.

Depending on the test type, more precisely the anchoring mode, the *pTcomponents* selected by the *Spattern* can be grouped into two categories:

- active - those *pTcomponents* that are considered processed (e.g. those that undergo a simplification operation). They will no longer be accessible for subsequent matching.
- witness - are the *pTcomponents* that have the role of context for the first category. Their absence invalidates the application of the pattern on the target sequence. They also remain available later for other patterns.

A *sTcomponent* can specify one of the following three types of testing - listed in the order of their specificity: word level, lemma level, part of speech level.

The current syntax of the rule description language does not allow multiple types of testing to be associated with the same pattern element. However, to alleviate this restriction, we can test the MSD compact label that includes all the features of the respective word (*pTcomponent*).

However, in a simple *sTcomponent*, the testing order is: lemma, word, part of speech. This seemingly unoptimized order is explained by the fact that a much larger number of *sTcomponents* use the lemma level testing.

In addition to structure tests, logical tests involving annotations are also applied. A *sTcomponent* may be required to have certain annotations in order to be matched.

Testing a set of components is a sequential operation, each simple component of the set being progressively tested, according to the procedure specified above, until a match is found or all options are exhausted.

Finally, testing an inner-pattern included in a *sTcomponent* involves a recursive call of the initial pattern testing method.

5.4 Preprocessing

This pre-processing operation is performed in order to improve the part of speech tagging operation. We observed that the result of this operation was negatively affected in direct proportion to the complexity of the analyzed sentence. This can be improved by making certain changes, which aim to preserve the structure of the analyzed sentence as best as possible.

The changes mentioned above can be grouped into the following three categories:

1. elimination of some constructions
2. adjusting certain expressions or words
3. reformatting punctuation marks

The constructions that are mentioned in point (1) can be included in the category of additional information, which is introduced through the parentheses, and therefore intervene in a negative way in the process of parts of speech analysis.

The phrases and words in point (2) are amended so that their parts of speech can be successfully identified. They will not be removed from the sentence, only a simplifying transformation operation is performed on them. This simple form is then provided to the labeling tool which will thus be able to correctly identify the part of speech. At the end of the process, their initial complex shape is restored.

Point (3) refers to certain punctuation marks that can be confusing. As examples we can mention the apostrophe characters used in proper nouns (e.g. D'Artagnan), or the period (".") and the comma (","), used in numerals (e.g. 23,564).

During the labeling stage, the tagger provides a series of information from which only the following are extracted:

- the word
- lemma
- the part of speech
- MSD special label
- The marking of noun phrases

Finally, as we mentioned, the original text is restored.

5.5 The stages of sentence simplification process

Obtaining the simple pattern, which corresponds to the canonical form of the sentence, involves the successive application of the steps specified in the MIXED rules file, accompanied by patterns that allow the identification of the linking words in LINKWORDS.

The steps for obtaining the simple pattern correspond to one of the rule categories defined in the rule configuration files. In order to be easily accessible, these categories are represented as tree-type structures having two types of nodes:

- control nodes - specify the information under them
- data nodes - contain a working *Spattern*

In the current implementation, two such tree-type structures are created. They are called *linkwords* and *mixed* after the name of the rule files from where the data is taken (figure 5-4).

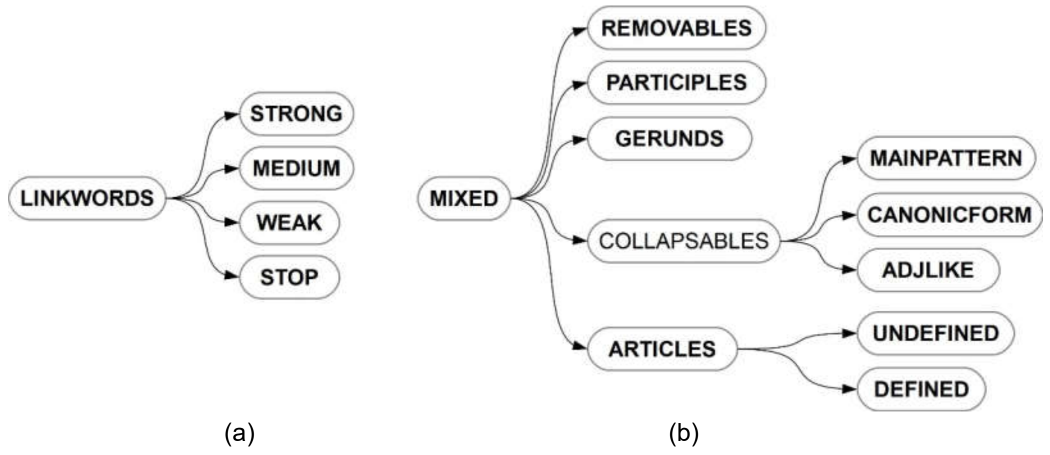


Figure 5-4 The structure of the rules configuration files

Figure 5-4 shows the main categories currently used. When creating, each node is associated with the path in the tree (a string) as shown in example Ex. 5.1. This path is stored in a list that can be easily queried.

Ex. 5.1 MIXED/COLLAPSABLES/MAINPATTERN

Both types of tree nodes are essentially identical having the same component data types, except that they are developed specifically for their role:

- control nodes are given a name and will have a list of subordinate control nodes and a list of data nodes
- data nodes have no names or subordinate nodes, instead each contains a pattern

5.6 Elimination of expressions without informational input

The stage uses patterns from the MIXED/REMOVABLES category.

This first stage involves the elimination of certain words and phrases that have no contribution, from an information point of view, in a definitional sentence. These are mostly adverbs or adverbial phrases used to underline or punctuate certain ideas (e.g. ”*also*”, ”*especially*”, ”*usually*”, etc.). Because they are not actually part of the text, they interrupt the natural sequence of words, having a negative impact on the analysis of the sentence.

5.7 Marking the linking words

This step uses all LINKWORDS patterns.

There are no directly defined patterns in the general category in the configuration file. All patterns reside in the STRONG, MEDIUM, WEAK and STOP categories that are included in it. However, there is a mechanism by which all the patterns in a category can be accessed, including those defined in subordinate categories. This fact is used at this stage, because we want to mark all the linking words, regardless of the category they belong to.

5.8 Marking adjectives with participle form

The category corresponding to this step is MIXED/PARTICIPLE.

This type of annotation does not necessarily refer to the part of speech in particular, but rather marks the role that this type of adjective plays in the sentence, in terms of meaning. For this reason, recognition rules have been created for certain words with a similar role (e.g. ”*capable* of”, ”*equal* to”).

In addition to checking the specific pattern, additional checking is needed to determine if the adjective has indeed the participle form, which is why a list of about 8000 words extracted from the dictionary of the labeling tool was created.

5.9 Marking conjunctions and punctuation marks

These are actually two distinct stages but, given that they are very simple and essentially identical, they will be presented together. Being so simple, none of them has an associated category of patterns, but instead the marking of the target words is done only by checking the part of speech. As punctuation marks we are only interested in the comma.

5.10 Marking of limit adverbs

In this step, certain adverbs that are part of noun phrase are annotated to help identify the points that may become the boundary between the noun that represents the definition super-concept and its attributes that are not considered important enough to be part it.

5.11 Preparatory simplification operations

This stage corresponds to the MIXED/COLLAPSABLES/MAINPATTERN category.

The patterns defined here are responsible for the intermediate simplification of noun attributes and are grouped according to the parts of speech that are mainly affected by them.

The purpose of applying patterns to a part of speech is to obtain a simplifying *pTcomponent* that includes all the words semantically connected to that part of speech.

Obs. Although we talked about the semantic connection, the information used to determine this aspect has actually a syntactic nature, which is why simplification decisions, although optimized as much as possible, can and will in some cases give erroneous results.

Adjectives

The first affected category of parts of speech is that of adjectives because they are the main form of attribute expression for both for nouns and for verbs with passive diathesis form.

Following each simplification operation, an adjective-type simplifying *pTcomponent* is obtained. The degrees of comparison, the simplification of adverbs related to adjectives are addressed here.

Verbs

The second group processed in this stage are the verbs. Apart from the verbs specific to the definitions, which are very precisely recorded, the rest are not very important, but still require a minimum of processing. In general, even if they appear in a sentence containing a definition, they do not end up being part of the extracted pattern.

The simplifications that apply to verbs are the consolidation of verbal forms that have several words: future tense, past tense, etc.

Articles

Another important group, that of articles, processes definite and indefinite articles.

Unlike the previous groups, although the attention is focused on this category of speech parts, the *Spatterns* will make the simplification by means of the *pTcomponents* of noun type.

Due to the fact that the indefinite articles disappear, it is necessary to annotate the resulting simplifying *pTcomponent* in a specific way, to preserve this information. Also, in order to have an uniformity and ease in processing, the nouns articulated with definite article are similarly annotated, even if this form being included in their structure is not lost.

Conjunctions

Because the processing of conjunctions is limited, the transformations described do not have a noticeable impact like those made so far. The sequences of interest are the comma followed by the conjunction “dar / but”, as well as the cases where the same conjunctions precede the conjunction “și / and”.

5.12 Marking limits in noun phrases

The pattern category associated with this step is MIXED/NPLIMIT.

This annotation stage is necessary to mark a semantic boundary within the noun phrases, which separates a portion of increased interest from the rest of the expression that is considered less important.

5.13 Annotating nouns according to the associated article

This stage consists of two sub-stages:

1. nouns articulated with indefinite article - MIXED/ARTICLE/UNDEFINED
2. nouns accompanied by the definite article - MIXED/ARTICLE/DEFINED

Given the fact that this stage has a well-defined and at the same time simple purpose, because the involved *pTcomponents* have already undergone a series of other preparatory transformations, the patterns in this category are very simple. Also, the annotations applied to the target nouns (definite / indefinite form) are implicit due to the fact that this stage was divided in two.

5.14 Generalities related to the final simplification operation of noun phrases

This step does not correspond to a specific category in the configuration file for the simplification rules. Being a very complex operation, it requires multiple iterations, as well as the preservation of state information between them. The final implementation of this simplification operation requires double-stranded recursion methods. Obviously this necessary complexity made it impossible to implement the operation using only the simple mechanisms offered by the language of pattern representation. As we have shown before it is centered on the pattern component and although some artifice could be done at pattern level, it simply does not have the means to transmit a state between two operations of pattern matching, and the recursion is totally out of question.

The simplification stage of substantive units consists of two sub-stages:

- detection of enumerations in the text
- simplification itself

In the sub-stage of detecting enumerations, the sentence is analyzed to find groups of nouns and adjectives that are similar in terms of the role they play. In other words, we want a grouping on semantic considerations. Their grouping allows these words to be treated as a single unit, which simplifies the subsequent analysis of the sentence.

The simplification sub-stage uses the information gained during the enumeration detection to progressively simplify first the noun phrases, and then the sentence itself, which leads to the final goal, reaching the canonical form and obtaining the simple pattern.

5.15 Enumeration detection stage

At this stage we want, as previously mentioned, the grouping of words with the same role. Because we do not actually have access to semantic information, we use any syntactic or lexical information that is available, trying to deduce as much as possible the role of words and the links between them, through the discovery of common patterns of use.

The speech parts of interest in the enumeration detection stage are:

- nouns – main elements around which the rest of the words are grouped
- adjectives – this is the stage in which adjectives will finally be simplified by incorporating in the noun to which they are connected
- commas – with an important role in controlling the process of creating enumerations
- specific conjunctions *and/or* – terminal control elements in enumerations

The method described here distinguishes between three types of enumerations:

1. enumerations of noun phrases
2. enumeration of adjectives
3. enumerations of adjectives derived from participle verbs

The distinction that is made between ordinary adjectives and those from participle is related to the fact that the latter have a specific role in the process of creating the simple pattern.

In addition to identifying enumerations, this method seeks to reassess the dependencies between nouns within the complex noun phrases. The analysis of all the noun phrases in the sentence thus constitutes a first step.

After this, a global analysis of the sentence is necessary to determine if there are enumerations formed by the main noun phrases. This second step is also performed using the enumeration detection method, but with a modified set of constraints.

Given a sequence of terms to be analyzed, the method of detecting enumerations aims to find the optimal matches between these terms to determine if their grouping is necessary and if so what is the best way to group them. For this purpose, several *candidate structures* will be generated reflecting these grouping variants and will be evaluated based on *reliability scores*.

In general, the method of detecting enumerations has the following steps:

1. go through the terms that must be analyzed until we meet a noun or adjective, thus constructing a work sub-sequence
2. launch the procedure of adding the respective sub-sequence - which adds a new main term
3. evaluate all the new variants in order to update the partial scores
4. resume from step 1 until all the terms in the sequence are exhausted
5. sort the final variants to find the optimal one

Reliability scores

The enumeration detection method recursively generates all possible candidate structures. which correspond to the possible ways of associating the terms from the input sequence, based on a set of constraints. Each candidate sequence is given reliability scores for a number of five categories:

1. *similarity score* - measures the degree of similarity between two nouns
2. *structural score* - evaluates the component parts of the enumeration and their arrangement
3. *discrepancy score* - is a cumulative handicap given to the enumeration for cases where there are suboptimal associations between nouns
4. *distance score* – it is calculated according to the distance in the sentence between the new noun and the one with which it is associated
5. *overall score* - is calculated by summing the scores of similarity, structure and discrepancy

The calculation of these scores is controlled by empirically developed numerical constants and boolean switches. In the current implementation there are two such sets of constants associated with the two distinct steps of the final simplification stage:

- the simplification of noun phrases step – the constants and switches are more permissive offering a greater freedom of association between terms
- the sentence simplification phase – certain restrictions are imposed to obtain noun enumerations with maximum degree of similarity

Appendix II lists the two groups of numerical constants.

1. **Similarity score**

The similarity score between two nouns is calculated taking into account a series of features of lexico-syntactic nature that are presented in table 5-1.

The final score ranges between 0 (total mismatch) and 100 (total match).

If the two nouns are accompanied by the same linking word, the similarity score receives a bonus, taking care not to exceed the maximum allowed limit.

Table 5-1 Comparison features for similarity score

Feature	**Values**
noun type	common, proper, unknown
linking word type	strong, medium, weak, stop, non-existent, unknown
article type	defined, indefinite, non-existent, unknown
noun case	direct (nominative-accusative), oblique (dative-genitive)

It is also important to note that this similarity calculation operation is not commutative.

$$SIMILARITY(noun1, noun2) \neq SIMILARITY(noun2, noun1)$$

2. **Structural score**

This score is iteratively calculated as items are added to the enumeration. To this end, we defined a series of rules that are contributing with partial positive or negative scores to the final value. Table 5-2 shows the set of rules as well as their value contribution.

Table 5-2 Rules for structure score

Nr. Crt.	**Rule**	**Value**
1.	If there are two or more conjunctions ***and***/***or*** of the same type	-100
2.	If there is both the conjunction ***or*** and conjunction ***and***	-2000
3.	If there is a comma without having one of the conjunctions ***and***/***or***	-50
4.	If there is a comma followed by a conjunction, a bonus is added for matches with more distant nouns	25
5.	If the enumeration contains a single noun separated by a comma from another enumeration whose compound type is very similar to said noun	-500

3. **Discrepancy score**

In the substantive unit detection phase, the labeling tool analyzes the words from left to right. Associations between words are made only on the basis of gender, number and case. A new term is always associated with the last term it matches, which is why in the resulting structure the new term is almost always included in the one before it, which can lead to association errors.

The idea is based on the fact that, in general, people tend to use in the same enumeration terms that are very similar to each other, this similarity being described by the similarity score.

The discrepancy score may be 0 in the case of a match that is considered optimal or have negative values, which increase as mismatches accumulate.

4. **Distance score**

In connection with the discrepancy score, a distance score is also calculated. It measures the distance between the index of the new item being added and the index of the one it is associated with. This score was introduced in the idea that if we have the same similarity to two other elements, the most appropriate element is the closest.

Procedure for adding a new term to the candidate structures

The terms that determine the launch of the addition process are nouns or adjectives. Everything else is treated as control elements (commas, conjunctions "*and*/*or*") or is of no interest.

The complex key elements in this structure in which the terms are added sequentially are the *enumeration* and *noun phrase* type. No simple element can be added other than as part of one of these two.

Nouns always mark the beginning of a new noun unit so they can be added immediately to an enumeration or as a subordinate element of another noun phrase. On the other hand, adjectives must be added to existing noun phrase elements.

The addition process is performed using a recursive search in the partial structure of enumerations and noun phrases created until then, because each new structure is included in a previous one, and it is generally necessary to reach the deepest "level".

A certain term can be attached in several ways within the structure, increasing each time the number of variants that will have to be explored in the next steps.

The addition is different depending on certain control factors, as well as the type of term to be added:

1. addition of adjectives - they can only be added to the noun they belong to, the dependence on the noun is not reevaluated. The structure is recursively traversed until the parent noun phrase is found, and the adjective is added at the end.
2. addition of nouns - there are three cases depending on the control elements (comma and conjunctions *and*/*or*) that accompany it.
   a. if there is neither a comma nor a conjunction - this means that the noun can only be a subordinate of another noun unit, and the addition is made in the parent noun unit (determined by the labeling tool).
   b. if there is both a comma and a conjunction - the noun will be added to an enumeration, but an indicator is activated stating that associations with “distant” noun phrases must receive a bonus.
   c. if there is only a comma or only a conjunction - the noun is added to an enumeration, without additional conditions or options.

Selection of the optimal candidate structure

At the end of the addition process, after analyzing all the terms, a list of candidate structures is obtained. They correspond to the possible variants of arranging the terms of the initial sequence, taking into account the present constraints.

To determine the optimal variant, we sort according to the scores presented above. In the order of importance they are:

- structure score – this score must be on the first position if we want to obtain a well-formed enumeration
- discrepancy score – following the tests performed, it was observed that the incorrect association of terms is a very important cause for obtaining suboptimal structures
- distance score – further helps to alleviate the shortcoming of incorrect association
- similarity score – helps to some extent to differentiate between variants.

Calculating the similarity score for enumerations

The similarity score of an enumeration reflects the overall fitting between its component noun phrases. To obtain this value, the similarity scores between any two noun phrases are first calculated, and then these values are averaged to obtain the similarity of the enumeration.

To effectively calculate the similarity of the enumeration it is necessary to have a list of all noun phrases as they appear in the enumeration. If the enumeration contains an inner enumeration, a special function will be called that creates a summarizing noun for that enumeration.

In this context, the similarity between two noun phrases NP1 and NP2 is defined as the arithmetic mean of all the similarities between any two consecutive noun phrases between NP1 and NP2.

Example Ex. 5.2 shows two enumerations. In the first enumeration (a) we notice that NP2 is a discordant part, which is quite broadly reflected in the adjacency scores and in the final score. In order to have a comparison standard, we provide the enumeration from (b) where the noun phrases are identical in terms of analyzed features.

We will only calculate the similarity for point (a):

$$Similarity_{NP1-NP3} = \frac{76.67 + 80}{2} = 78.34$$

Ex. 5.2

```
a. cartea, creion și caietele
   NP1    NP2     NP3
      76.67    80
b. cartea, creionul și caietele
   NP1    NP2       NP3
     100      100
```

Creating summary nouns

A summary noun concisely expresses the features of an enumeration. It has the dominant characteristics of the component noun phrases.

Each feature is determined by analyzing that particular type of feature in each noun phrase, choosing the value with the majority number of occurrences. If this majority is not reached, the value from the first enumeration element will be chosen. This element is considered the most important because it gives the enumeration a certain distinct imprint.

5.16 The final simplification step of noun phrases

Within this stage, the analysis made in the enumeration detection phase, is put into practice. The optimal variant of term association obtained there describes how to finally simplify the noun phrases and the sentence. This stage is also based on a recursive process that dives into the structure until it reaches the bottom elements, and then starts to progressively simplify them from the bottom up.

Processing enumeration elements

Each simplified enumeration must be assigned a symbolic linking word chosen based on the linking words accompanying the noun phrases that compose it. For this purpose we use a procedure that determines the type of linking word with maximum influence.

Table 5-3 shows the values of the coefficients associated with each type of linking word.

Table 5-3 Coefficients of linking word types

Linking word	**Coefficient**
STRONG	1
MEDIUM	-0.6
WEAK	-0.4
STOP	-1000

The linking words belonging to MEDIUM type, to which the oblique cases (genitive/dative) correspond cannot appear together with the types corresponding to the direct cases (nominative/accusative). In practice, the only types of linking words that can be combined in an enumeration are STRONG and WEAK. Due to its usage patterns, the STOP type cannot also appear in an enumeration along with other types.

The values in the table were chosen according to the following principles:

$$|1S| > |2W|$$

$$|1S| > |1M|$$

Obs.

1. In the above relations S, M, W are the initials of the STRONG, MEDIUM and WEAK types.
2. The type STOP that must determine the immediate transition is not taken into account, having, as can be seen from the table, a considerable value to force this.

Given a series of coefficients c_i cu $i = \overline{1,n}$ the calculation formula for the score corresponding to this series is:

$$Score = \sum_{i=1,n} c_i \left(1 - \frac{i-1}{n}\right)$$

This formula aims to give more importance to the linking words from the beginning of the series (the first having an unaltered coefficient), importance that gradually decreases as we move towards the end.

Processing noun phrase type elements

The processing of noun units has a high degree of complexity because it consists of three phases:

1. simplification of the adjectives from participle

2. simplification of the adjective from participle enumerations
3. simplification of the noun phrase

The order of these operations cannot be altered because the result obtained in a certain phase is necessary in the subsequent phases.

1. To obtain the simplifying *pTcomponents* of the adjectives from participle, we are searching the vector of elements for sequences of words with the following parts of speech:

- ADJECTIVE (PARTICIPLE) NOUN
- ADJECTIVE (PARTICIPLE) APPOSITION NOUN

2. Finding the enumerations of adjectives from participle involves going through the vector of elements in search of a first such adjective. When it is found, another search is launched aiming to extract a possible list starting from that location. The search will be performed as long as proper adjectives, commas or conjunctions *and*/*or* are found. It should be noted that in this phase of execution we can only find adjectives from participle or the simplifying adjectives of the sequences processed in the previous phase.

3. Finally, it is possible to perform the last phase which involves the creation of the simplifying *pTcomponent* for the noun phrase. The label of the corresponding simple pattern component is also created now. This a string that briefly describes the main features:

- article type – is considered the type of the pattern component
  - NPDEF (definite article)
  - NPUNDEF (indefinite article)
  - NP (without article)
- noun case – the suffix “OBL” (e.g. NPUNDEF_OBL) is added at the end for oblique (genitive/dative) cases. Direct forms are implicit, so the pattern component remains unchanged.

5.17 Storing simple patterns

This process involves the creation of lists of patterns grouped into several categories that correspond to the formal units of definitions: RGET, VERB, GENUS, REST.

For TARGET and HYPER units this procedure does not apply for two reasons:

- they are always composed of a simple noun or a noun phrase, which is why their analysis would bring absolutely no extra resolution, because in the end we get only one pattern for each unit
- it would divide the RGET and GENUS units into two parts, increasing the complexity of processing and analysis them

Instead, their positions within the formal parent units are extremely important and are marked so that they can be used at the right time.

Within each formal unit category, the patterns are additionally grouped according to the shape of the simple pattern to reduce the number of checks required for testing.

Listing 5-1 shows an example of a simple pattern for the GENUS formal unit and some of the corresponding definitions sequences taken from the training corpus.

On the first line are the components of the simple pattern to which have been added, in order to facilitate their understanding, a postfix that contains the abbreviation of the formal unit of the definition of which they are part.

For simplicity, we will refer to these simple pattern components by the generic notation *Tcomponent*.

NPUNDEF_hyp	care_gen	VERB_gen	ADPOZ_gen	NPDEF_gen
o componentă software	care	rulează	în	contextul alt unui program
un program de calculator	care	ajută	la	pregătirea unui microprogram
o specie de broască de copac	ce	trăiește	în	estul Australiei
un tip de speleothem	care	atârnă	de~pe	tavanul sau zidul peșterilor de calcar

Listing 5-1 Simple pattern

The division into sub-patterns is done by grouping the sequences of *Tcomponents* according to the formal units they belong to. If certain *Tcomponents* extend between two consecutive units, they will be introduced in the first one.

Obs. Belonging to two formal units occurs due to the fact that unlike the automatic process of simplifying the sentence that follows a series of strict rules, the action of annotating the formal units in the corpus has a much more organic nature that takes into account the meaning of words.

In this context, two simple patterns are considered identical if they have the same number of components and all corresponding *Tcomponents* match. It should be noted that the match between the components does not imply that they are perfectly identical.

When the simple pattern already exists in the pattern database, we only try to add the new text of the definition among the existing ones. For this, another extremely strict check is made, this time at the term level, in order to prevent the addition of an identical text.

Simple pattern component

This component determines, depending on its type, how the corresponding term in the analyzed phrase is viewed. We will present below these types accompanied by a series of explanatory comments:

- class of direct nouns (nominative/accusative case) (NOUN, NP, NPDEF, NPUNDEF, NP_ENUM, NPDEF_ENUM, NPUNDEF_ENUM)
- class of oblique nouns (genitive/dative case) - same as direct ones, but with the postfix "_OBL" added (e.g. NPDEF_OBL)
- class of adjectives from participle (PARTICIPLE, PERTICIPLE_ENUM)
- morphological information class (WORD, LEMMA, POS_TYPE)
- LINKWORD_TYPE - special type for marking keywords. In addition to the simple elements, it contains the type of the linking word.
- CUSTOM - special type for dealing with cases where the pattern element is imposed by the annotation mechanism

Obs.
1. NOUN and NOUN_OBL are intermediate simple pattern elements. They do not appear in the final form of the pattern.
2. In the case of noun classes the form for enumeration is considered identical to the simple form of the same article type (e.g. NPDEF == NPDEF_ENUM). We will consider these related types to be part of the same extended type. The same is true for adjectives from participle.

5.18 Concluzii

Contribution list for chapter 5:

- creating a support for the internal representation of the patterns corresponding to the analyzed sentences and implementing the necessary handling operations
- creating an engine for interpreting and comparing simplification patterns and sentence patterns
- implementing the algorithms that use the simplification rules defined in the configuration files
- developing a way to identify and simplify enumerations in the text

This chapter describes the algorithm by which sentences are analyzed and processed, preparing them for definition detection phase. The result of sentence processing is a pattern that is as simplified as possible, but detailed enough to be able to make the distinction between common sentences and definitions. It is called the *simple pattern* of the sentence.

The process of analyzing a sentence goes through several stages, namely: preprocessing, labeling, simplification and finally the creation of the simple pattern.

The first of the above steps involves preprocessing the text at both morphological and syntactic levels to help the labeling tool make the best decisions in certain difficult cases. For this, a temporary preprocessing is applied before labeling.

The simplification stage has a very high complexity and involved the creation of algorithms and routines necessary for sentence processing.

During the analysis, the sentence is stored in a lattice of pattern components, which is able to preserve all the successive stages of simplification. It allows the use of operations such as: adding, removing, repositioning, simplifying nodes.

To test the sentence simplification patterns defined in the three configuration files, a specific interpretation and comparison engine has been created which successively tests the possibility of applying each pattern. In the case of a successful matching operation, all annotation and simplification operations present in the pattern are applied to the corresponding terms in the sentence.

The sentence simplification contains several sub-stages such as: the elimination of certain unimportant expressions in the context of definition analysis, the marking of linking words, the marking of certain adjectives with participle form, intermediate simplification operations, etc.

The most important sub-stage is the final simplification of noun phrases that can reduce complex term structures to a single noun. In parallel with this operation, the enumerations of terms within the noun phrase are identified. To this end, a set of rules based on numerical constants and boolean switches has been developed, which controls the mechanism for term association. The choice for the optimal combination variant is made by using reliability scores that analyze (a) the similarity between nouns, (b) the structural elements of the enumerations, (c) the degree of discrepancy and (d) the distance between two associated nouns.

The last step of creating the patterns involves dividing the simple pattern obtained after simplification according to the formal units of the definition, resulting in sub-patterns for RGET, VERB, GENUS, REST. Each of these sub-patterns will be added to a specific list.

6 Definition detection - Experimental validation

The process of definition detection consists of the following steps:

1. processing the sentence to be tested in order to create the associated simple pattern
2. testing with the sub-patterns created for the formal units of definition

In the first stage, the procedure for creating simple patterns is common to both the training and the testing phase, so we apply the same processing operation used during training.

In the second stage, the aim is to find a set of sub-patterns that will cover as extensively as possible the text of the sentence. We are talking about a set of sub-patterns because they do not have to come from the same definition in the training corpus.

The use of sub-patterns for each of the formal units of the definition has the following advantages:

- we improve the ability to detect definitions by increasing the coverage of term configurations that may appear in a definition
- it becomes possible to identify the formal units of the definition, which facilitates the marking of the concept and the super-concept within the definition

6.1 Case study

A collection of texts from the Encyclopedia of Science [61], a book containing information from various fields of the scientific world, was used to test the classifier. This collection of texts consists of 1128 sentences, of which 112 are definitions, covering a number of 17 topics, each grouping around 50-60 sentences.

In order to process the texts on several levels, each of the 17 topics was considered a document so that we could perform both a separate analysis of each and a general analysis of all.

This study was conducted with two objectives in mind:

1. first of all testing the performance of the classifier
2. the use of test results to study a possible correlation between the terms constituting the concepts and super-concepts in the definitions, on the one hand, and their relative frequency (or TF-IDF).

More precisely, the aim was to confirm the hypothesis that if we take all the concepts from a document and order them according to the frequency values (or according to TF-IDF), the most important concepts and super-concepts from the definitions will be among the first.

For brevity, below, we will use the phrase ***definitional concept*** to refer collectively to the terms that in definition constitute the defined concept and its super-concept.

Because the defining concepts can be easily obtained using the markers provided by the classifier, we will talk about identifying the rest of the concepts in the document.

These are groups of up to three words extracted from the sentences, using a certain set of rules that aims to obtain only valid concepts, as opposed to extracting random groups of consecutive words.

In this context, by valid concepts we refer to entities expressed through nouns that may be accompanied by possible attributes.

The rules governing the creation of these word groups are as follows:

- it is mandatory to have at least one noun in the sequence
- any other word in the sequence other than the main noun must logically belong to it or help to connect a word that logically belongs to it

6.2 Classification results

After analyzing the sentences in the test corpus, the results presented in the confusion matrix in table 6-1 are obtained.

Table 6-1 Confusion matrix

		Real	
		Definition	Non-definition
Classification	Definition	72	40/16
	Non-definition	27	987/1010

The table shows that only 72 of the 112 definitions were correctly identified by the classifier. In addition, a number of 27 common sentences are classified as definitions.

Testul 1	Testul 2
TP (true positives) = 72	
FN (false negatives) = 40	**FN** (false negatives) = 16
FP (false positives) = 27	
TN (true negatives) = 987	**TN** (true negatives) = 1010

The measures that describe the classifier performance are presented below:

a. **Precision** (true positive rate) – measures the proportion of correctly classified definitions out of the total sentences that have been classified as definitions
$$Precision = \frac{TP}{TP + FP}$$
b. **Recall** (true negative rate) – measures the proportion of correctly classified definitions out of the total definitions
$$Recall = \frac{TP}{TP + FN}$$
c. **Accuracy** – measures the proportion of correct predictions of the classifier
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$
d. **F1-score** – another measure of test accuracy that provides a single score based on the values obtained for accuracy and recall, representing the harmonic average of the two
$$F1\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2TP}{2TP + FP + FN}$$

Using the above formulas we will obtain the following values for the four performance measures:

Testul 1	Testul 2
$Precision = \frac{72}{72+27} = 0.73$	$Precision = \frac{72}{72+27} = 0.73$
$Recall = \frac{72}{72+40} = 0.64$	$Recall = \frac{72}{72+16} = 0.82$

$Accuracy = \frac{72+987}{72+987+27+40} = 0.94$ $Accuracy = \frac{72+1010}{72+1010+27+16} = 0.96$

$F1\text{-}score = \frac{2*72}{2*72+27+40} = 0.69$ $F1\text{-}score = \frac{2*72}{2*72+27+16} = 0.77$

6.3 Analysis of classification errors

We will aim to analyze notable cases for each class of errors to determine the causes of their occurrence. Each type of error will be accompanied by a series of representative examples.

False positive cases

❖ From a semantic point of view, certain phrases are situated at the limit where they can almost be considered definitions. For some of them we could even argue that they could have been marked as definitions in the manual annotation phase.

Ex. 6.1

a. Forța râului este un curent mai puternic. / The force of the river is a stronger current.
b. Majoritatea antenelor sunt tije, baghete sau discuri metalice. / Most antennas are rods, wands or metal discs.
c. Lumea naturală este un loc amestecat în care se combină tot felul de materiale. / The natural world is a mixed place where all kinds of materials are combined.

❖ A series of sentences, although syntactically in the form of a definition, can only be correctly identified using semantic information. This difficulty is introduced, in general, by using the definite verb *to be*, widely used in the most diverse semantic contexts.

Ex. 6.2

a. Punctul de sprijin este piesa din mijloc, pe care stă balansoarul. / The point of support is the middle piece on which the rocker stands.
b. Iarna este momentul ideal pentru a identifica steaua Sirius. / Winter is the ideal time to identify the star Sirius.

❖ Some potential definitions are incomplete because they partially or totally lack the defined concept (*definiendum*). They define a term located in one of the previous sentences, which is referenced, for example, using a pronoun.

Ex. 6.3

a. Această rată este frecvența lor. / This rate is their frequency.
b. Această pădure vastă este numită pădure boreală sau taiga. / This vast forest is called the boreal forest or taiga.

False negative cases

❖ A very large number of cases, perhaps even the majority, occur due to the fact that certain terms are misidentified by the labeling tool and are assigned a wrong part of speech.

Ex. 6.4

a. ***Pana*** este o mașină simplă folosită pentru tăiat.
   – correct: **noun** - pană/wedge – wrong: **preposition** - până/until
b. Undele radio sunt ***unde*** lungi, invizibile, cu energie joasă.
   – correct: **noun** - unde/waves – wrong: **adverb** - unde/where
c. Un aerogel este un ***solid*** cu un gaz dizolvat în el.
   – correct: **noun** - solid/solid – wrong: **adjective** - solid/solid

❖ Another category of errors occurs due to the fact that the definition pattern does not exist, because there were no definitions with a similar format in the training corpus. This is due to the fact that many definitions in the test corpus are very far from the classic Aristotelian pattern that characterizes the training set.

Ex. 6.5

a. Prima este *lungimea de undă* **-** *distanța dintre un vârf al undei și altul.* / The first is *the wavelength* **-** *the distance between one peak of the wave and another.*
b. *ANTIBIOTIC* **:** *medicament care ucide bacteriile* sau le împiedică să se reproducă. / *ANTIBIOTIC* **:** *a medicine that kills bacteria* or prevents them from reproducing.
c. *Locurile unde masele de aer diferite se întâlnesc* ***se numesc*** *fronturi*. / *The places where different air masses meet* ***are called*** *fronts.*

6.4 Studying the correlation between definitional concepts and frequency (TF-IDF)

As mentioned before, the working hypothesis related to the important definitional concepts is that they have a higher relative frequency in the document from which they come, than most other concepts.

In this study we are interested in defining a method for automatically identifying these important defining concepts.

For this we will study the results obtained using relative frequency (TF) compared to the TF-IDF measure of the respective terms, to determine if one of them has a greater power of discrimination.

The definition of the IDF TF measure is the following:

> If a collection of documents *D* is given, TF-IDF is the statistical measure that allows the assessment of a term importance for a certain document *d* from collection *D*.

To determine the percentage of important definitional concepts in the document, we propose two algorithms whose results will be analyzed comparatively to determine which is most suitable for the purpose.

We must specify that many stages are common to both algorithms, the difference being registered in the selection of important definitional concepts. For this reason we will present in full only the first algorithm, and for the second we will specify only the differences.

❖ *Algorithm 1*

1. the records in the table will be sorted by *docID*, *TF* (or *TF-IDF*) and finally by expression
2. for each document,
   2.1. we count the TP type records (corresponding to the concepts from the correctly identified definitions). We will refer to this value by nr_{DC}
   2.2. will count how many of the first nr_{DC} records in the list are in fact of TP type. This value, representing the number of important definitional concepts, will be denoted by nr_{ImpDC}
   2.3. we calculate the ratio between nr_{ImpDC} and nr_{DC}, which gives us the percentage of important definitional concepts
3. finally, in order to calculate the overall percentage of sufficiently important definitional concepts, we will calculate the arithmetic mean of the partial values obtained for each document.

Obs. If a document does not have any definitions, meaning that nr_{DC} is 0, the result of step 2 is automatically set to 0.

❖ *Algorithm* 2

Step 1 is slightly modified because only the TF will be chosen for the second sorting criterion.

Step 2 which is dealing with the selection of important concepts is different, as it can be seen below:

2. for each document
   - 2.1. we count the TP type records denoted by nr_{DC}
   - 2.2. we calculate the average value for the relative frequencies of all the concepts, which will be denoted by nr_{avg}
   - 2.3. we count how many of the defining concepts have a relative frequency higher than nr_{avg}, a value that will be designated as nr_{ImpDC}
   - 2.4. we calculate the percentage of important definitional concepts nr_{ImpDC} / nr_{DC}

In case of algorithm 1, the reasoning behind choosing the total number of definitional concepts nr_{DC} as a limiting factor for each document is based on two considerations:

1. the number cannot be a constant, it needs to be adaptable to the number of existing definitions in the document
2. if all the definitional concepts were important, they would be placed entirely on the first positions in the table, which leads us to conclude that nr_{DC} is a reasonable interval to choose the important ones

The values of the percentages obtained for the 17 documents (ratios nr_{ImpDC}/nr_{DC}), as well as the final score (arithmetic mean) are shown in table 6-2.

Table 6-2 Percentage of definitional concepts in test corpus documents

doc	TF(alg1)	TF-IDF	TF(alg2)	doc	TF(alg1)	TF-IDF	TF(alg2)
1	0.4	0.4	0.45	10	0.1	0.2	0.6
2	0.57	0.57	1	11	0.14	0.29	0.57
3	0.33	0.33	1	12	0.22	0.33	0.56
4	0.36	0.36	0.45	13	0.22	0.22	0.33
5	0.33	0.33	0.67	14	0.14	0.29	0.71
6	0.22	0.22	0.61	15	0.18	0.18	0.47
7	0.36	0.45	0.45	16	0	0	0.43
8	0.25	0.32	0.39	17	0.2	0.2	0.6
9	0.6	0.6	1	**scor**	**0.27**	**0.31**	**0.6**

In the case of *algorithm 1* we consider that the TF-IDF variant is better because it is able to place more of the definitional concepts in the range designated to the important ones. The final overall results obtained by arithmetic mean show that in general 27% (respectively 31%) of the definitional concepts can be considered important.

Although *algorithm 2* is able to select several definitional concepts, it can be perceived as too permissive because it accepts concepts with a relatively low relative frequency compared to those at the top of the hierarchy, many of which are not definitional concepts.

For these reasons, we chose to use the first algorithm, the results of which we will evaluate in the next section.

6.5 Results validation

Because we do not have an automatic way to determine the importance of concepts that is essentially a semantic analysis operation, we will use an intuitive human assessment method to determine how representative the selected definitional concepts are for the source document.

To give an example, we will consider the important definitional concepts extracted from documents #1 and #3, which are presented in table 6-3.

Table 6-3 Important definitional concepts

docID	concept	nr	percent
1	forță / force	29	8.71%
	mașină / machine	19	5.71%
	mașină simplu / simple machine	8	2.40%
	rampă / ramp	7	2.10%
	forță active / active force	6	1.80%
	pană / wedge	6	1.80%
	șurub / screw	5	1.50%
	tip / type	4	1.20%
3	undă / wave	29	13.94%
	undă sonor / sound wave	4	1.92%

Going through the list we can immediately realize that document #1 talks about a series of simple devices (*mașină simplă* / *simple machine*, *rampă* / *ramp*, *pană* / *wedge*, *șurub* / *screw*), as well as how they interact with the environment (*forță* / *force*, *forță active* / *active force*).

For document #3 we notice that the percentage of occurrences of concept *undă* / *wave* out of the total number of occurrences for all concepts is very high (almost 14%), which helps us to deduce with high confidence the general theme of this document.

An interesting case is the situation of document #16 which apparently does not have important definitional concepts, as can be seen in the data presented in table 6-3. In fact, this document also contains important definitional concepts such as *stea* / *star* (37 appearances - 10 %), *soare* / *sun* (6 appearances - 1.62%), *stea neutronică* / *neutron star* (5 appearances - 1.35%), *fuziune* / *fusion* (4 appearances - 1.08%) which, however, appear in atypical definitions, which is why they have not been recognized as such by the definition classifier.

Starting from the premise that there may be undetected atypical definitions, based on concepts with high TF-IDF, we can make suggestions for possible defining contexts that include these concepts, which can be subsequently validated in a supervised manner.

In conclusion, we consider that these defining concepts have a sufficiently high capacity to describe the topics present in the documents of origin. This gives them, in our opinion, a sufficient degree of importance, thus validating the selection method presented in *algorithm 1*.

6.6 Conclusions

Contribution list for chapter 6:

- implementing the definition testing algorithm

- studying the connection between the definitional concepts and the frequency (alternatively TF-IDF) in the document
- developing a method to automatically determine the important definitional concepts

This chapter presents the procedure for testing a sentence for the purpose of detecting definitions. A successful identification involves finding a valid combination of RGET, VERB, GENUS and partially REST sub-patterns that match the terms of that sentence. An important feature of this operation is that these sub-patterns can come from different definitions in the training corpus.

In the last part of the chapter we make a case study for testing the classifier, using a collection of 1128 phrases, covering a total of 17 topics (documents) and containing 112 definitions. In parallel, we aim to determine whether important concepts and their super-concepts from definitions have a higher relative frequency and/or TF-IDF measure than most other concepts from the documents to which they belong.

Following the testing of the classifier for all the definitions in the corpus we obtained a precision of 0.73, a recall of 0.64, an accuracy of 0.94 and a F1-score of 0.69. Considering only the Aristotelian definition set we get a precision of 0.73, a recall of 0.82, an accuracy of 0.96 and a F1-score of 0.77.

The causes of classification errors for false positive and false negative cases are also analyzed, providing actual examples for each of them.

To validate the working hypothesis that places important definitional concepts at the top of the hierarchy of all concepts in a given document, the relative frequency (alternatively the TF-IDF measure) was used as the ordering criterion. The selection of important concepts is made observing that if all the definitional concepts were very important, they would be placed among the first N positions of the list. To obtain a representative value for each document, N is chosen as the total number of definitional concepts extracted by the classifier. Next, we can determine the percentage of important definitional concepts by counting how many are actually among the first N positions. Finally, calculating the arithmetic mean of all values, we obtain the general percentages of the selected definitional concepts, which are 27% for the relative frequency and 31% for the TF-IDF measure.

7 General conclusions, original contributions and future perspectives

Through the research carried out in this thesis we aimed to develop for the Romanian language, a method of extracting definitions from texts, mostly from the structured and semi-structured texts. Being specially designed for maximum compatibility with the Romanian language, the method takes into account its particularities for optimizing the performance.

In order to gain an overview of the field of definition detection and extraction, we analyzed previous studies in this field to discover the advantages and disadvantages of each method, in order to find out which of them would be most suitable and which could be adapted for our purpose.

In this direction, chapter 1 reviews the methods for detecting/extracting definitions, which can be placed in two main categories.

In the first category we have methods that identify definitions using **lexico-syntactic patterns**. Manually creating these patterns is an unsophisticated way of obtaining them, that requires direct text analysis following an iterative process. They can also be acquired automatically using a specifically annotated training corpus or, alternatively, the bootstrapping technique.

These patterns can either be created manually through direct text analysis following an iterative process, or automatically involving the use of an annotated training corpus or, alternatively, the bootstrapping technique.

The second category includes methods which implement **machine learning techniques**. Training their models requires various features which are usually created based on the lexico-syntactic properties of the definitional sentences.

They use for training various features, usually created based on the lexical and syntactic properties of the definitional sentences.

The chapter also presents the preprocessing techniques of the text collections used for training and testing (section 1.3) and important methods of definition analysis (section 1.4).

The thesis contributions are presented in chapters 2-6, which describe the process by which a method for extracting definitions for English, based on lexical syntactic patterns, was modified for Romanian, along with its training corpus of definitions. In the end, the main processing engine of this technique was completely overhauled because it proved inefficient for the Romanian language. The training corpus was translated into Romanian keeping the essence of the initial manual annotation, but adapting it as necessary.

The classifier described in this paper was written in the Java programming language and implements all the ideas described in these chapters.

Chapter 2 presents the changes to the original method of extracting definitions in order to adapt it to the specifics of Romanian language.

As we mentioned before, the main mechanism of automatic pattern creation was almost completely reworked. The English method uses so-called **star patterns** created based on word classes. The class can be the word itself, if it has a high frequency (e.g. prepositions), otherwise it is the part of speech. Through this operation the pattern acquires a certain generalization degree. For the Romanian language, where the use of prepositions within noun phrases (e.g. ”obiect ascuțit *în* formă *de* pin” / ”pin-like sharp object”) is very high, the patterns obtained this way would have had a very low capacity for generalization. As an alternative, we simplify the phrases according to certain criteria, so that the formal units of the definition, especially the part that constitutes the explanation, will have the optimal ratio between the informational importance of the word sequence and the number of words.

This ratio is not actually calculated, but rather estimated by introducing the concept of **linking words**. Every such word, depending on its characteristics, has the tendency to introduce information with a certain specific importance degree. The linking words are described in detail in section 2.4 and are grouped into four categories that reflect their informational importance (STRONG, MEDIUM, WEAK, STOP).

The categories of linking words are governed by default rules based on which the manual annotation of the formal units of the definitions from the training corpus was performed. A good number of particular situations related to certain linking words have led to the creation of special rules.

In order to gain an understanding of the usage patterns of linking words, a series of measures have been defined, describing their features: **fitness**, **mobility**, **predError** (prediction error) which are calculated based on statistical numerical data extracted from the training corpus: *number of occurrences*, *number of transitions*, *number of exceptions*. This study is presented in section 2.5.

In the first part of chapter 3 we present a case study on the characteristics of five tools for labeling the parts of speech that are available for the Romanian language, in order to choose the most appropriate one.

Section 3.2 presents in detail the chosen instrument, UAIC Pos Tagger, which, having a hybrid nature, allows us to adjust the results using directed graphs-type rules. These rules are in fact patterns that can operate adjustments on the parts of speech labels, if they match a certain sequence of terms in the analyzed sentence. Also, the same system of rules is used in chunking to mark the boundaries of noun phrases.

Our contribution to this process was to improve and add new rules for labeling parts of speech, thus correcting a number of common errors, as well as improving the rules for marking the noun phrases.

Chapter 4 describes a language specially designed to represent simplification patterns, which are extremely important mechanisms of the method. They are used to sequentially simplify the analyzed phrase. The language syntax has been designed so as to allow an easy definition of the simplification rules, facilitating their creation and debugging.

These patterns are defined by means of three configuration files each with a certain specificity, and section 4.3 presents the structure of these files and their general syntax.

In the second part of the chapter, section 4.5, describes the syntax elements related to the individual pattern components. This syntax is component-centric, because the operations specified by the syntactic operators apply exclusively to the corresponding term in the analyzed sentence.

There are control components that influence the structure of the pattern and structural components that are actually used to test the matching with the sentence terms. Each structural component defines the type of testing (word, lemma, part of speech level) that will be performed and can be accompanied by a series of modifiers that influence the testing procedure (e.g. recurrent modifier, optional modifier, etc.).

Chapter 5 describes the steps for creating the simple pattern for the analyzed sentence.

Section 5.2 shows the internal representation of this pattern. To allow the non-destructive simplification of the sentence we developed a mechanism based on a directed graph and defined a series of atomic operations: **simplification** of a sequence of nodes, **removal** and **addition** of nodes, **interchanging** of nodes, **repositioning** of a node.

From now on we will refer to the pattern corresponding to the analyzed sentence by the name of **processing pattern** to differentiate it from simplification patterns.

To help the labeling tool provide the correct labels for the difficult terms in the sentence we perform a temporary simplification operation which is described in section 5.4.

Starting with section 5.5, we talk about the steps for obtaining the simple pattern.

Section 5.14 outlines the simplification of noun phrases, which consists of two stages:

- identifying the enumerations - in which the analysis of each noun phrase is performed
- the actual simplification

The enumeration step is presented in section 5.15. In this phase, we want to identify the enumerations of nouns, adjectives and, in particular, adjectives from participle (they have a special role in creating the simple pattern). At the same time, the aim is to re-evaluate the relationships between nouns, using certain similarity aspects. Finding the best configuration is based on reliability scores, which are as follows: **similarity**, **structure**, **discrepancy** and **distance** score.

The result obtained in this analysis phase is used for the actual simplification of the noun phrases which is described in section 5.16. The process is recursive, starting with the inner-elements, progressively operating simplifications of inner enumerations or noun phrases, as required.

Finally, in section 5.17, we present the method of storing the simple patterns. To maximize the coverage of patterns, they are divided into sub-patterns corresponding to the formal units of the definition. This way, when tested, new complete patterns can be generated using sub-pattern combinations, regardless of their origin.

The first part of chapter 6 describes the method of testing sentences to determine whether they are definitional or not. For this we use a test corpus composed of 17 documents, each having around 65 sentences, the text type being semi-structured. The total number of sentences is 1128, out of which 112 are definitional.

In section 6.2 we present the classification results. Performance is assessed by calculating the usual indicators: *precision*, *recall*, *accuracy*, *F1-score*. Here, we also discuss the most common causes for *false positive* and *false negative* cases.

The second part of the chapter uses the extracted definitions to explore the hypothesis that important concepts corresponding to the defined term and its super-concept (within the definitions) have an increased frequency in the document. In the study these concepts are called **definitional concepts**.

We must mention that by important concepts we refer to those concepts that are very representative of the text to which they belong to, and could possibly be used in a summarization process.

In section 6.4 we present two algorithms for automatic selection of important definitional concepts whose results are comparatively analyzed. In section 6.5 we validate, through noteworthy examples, the choice of the optimal algorithm, presenting the interesting results obtained from its application.

7.1 Contributions

The following are the main contributions of the thesis:

Chapter 2

- adapting the annotation of formal units of the definitions for Romanian language
- defining the concept of linking word
  - progressive experimental establishment of linking words characteristics
  - establishing the four categories: STRONG, MEDIUM, WEAK, STOP
  - establishing the basic rules of formal units annotation for each category
- developing an extensive set of annotation rules for dealing with linking words exceptions
- developing measures describing the use patterns of linking words which highlight these exceptions
- identifying a new method for generating testing patterns - **simple patterns** - based on the simplification of noun phrases

Chapter 3

- improving the performance of the morpho-syntactic analysis tool
  - improving the rules for correcting parts of speech tags
  - improving the rules for marking noun phrases

Chapter 4

- establishing the structure of sentence simplification patterns
- determining the general structure of configuration files for the simplification patterns
- creating a language for representing simplification patterns
  - establishing the general syntax of the files
  - establishing the syntax of the simplification pattern components
  - defining two types of components - *control* and *structural*
  - developing a set of modifiers for structural components

Chapter 5

- creating a mechanism for the internal representation of the patterns corresponding to the analyzed phrases
  - implementation of atomic operations necessary for pattern processing
- development of a processing engine for comparing the simplification patterns with the sentence patterns
- simplification of sentences to obtain simple patterns
  - elaboration of algorithms specific to each simplification phase, using the patterns defined in the configuration files
- development of a way to identify and simplify enumerations
  - elaboration of reliability scores to help evaluate each candidate enumeration structure

Chapter 6

- implementation of the sentence testing algorithm for definition detection
- testing the developed classification tool
- conducting a study on the connection between the definitional concepts and their frequency (respectively TF-IDF) in the document

- elaboration of an algorithm for automatic selection of important definitional concepts

7.2 Future directions

We will present below the ideas on possible research directions in order to increase the performance of the classification tool developed in this doctoral thesis.

A relatively easy-to-achieve method that can improve the classifier performance by reducing false negative cases would be to introduce into the training corpus new definitional forms for which there are no patterns. Although currently the training corpus contains exclusively definitions that conform to the classical Aristotelian pattern, it is also possible to analyze some forms that deviate slightly from this pattern, for example the glossary type in which the defined term and explanation are separated by a character, usually "-" or ":".

Another possibility for improvement is to deal with cases where the defined word is at the end of the sentence. This type of definition should be based primarily on verbs that are specific to definitions such as "*to be named*", in order to eliminate as much as possible the introduction of false positive cases, as in the case of general verbs such as "to be". The main problem that arises here is to correctly determine the beginning of the definition, more precisely the part of the definition that contains the explanation of the defined term. Being placed before the definitional verb, we no longer have the luxury of knowing exactly where it begins, because the sentence may contain at the beginning a sequence that is not part of the definition.

Cases where the definition extends between two sentences can be addressed to some extent, but require major changes in the current way of analyzing. Also, given the uncertainty of properly identifying the parts of the definitions, the use of a degree of confidence should be introduced for them.

At present the result of the classification can only be *definition* or *non-definition*. It would be useful to refine the classification mechanism to allow all definitions to be assessed in terms of a degree of confidence.

In its current form, the classifier can only extract a single definition from a definitional sentence. Although the annotation of formal units in the training corpus deals with these cases, the current analysis engine does not integrate the detection of cases in which a certain term has several explanations in the same definitional sentence. The problem in this case is to determine the points of separation between the explanations.

The procedure for extracting the definition can also be refined, because at this moment the noun phrases corresponding to the explanatory part of the defined term are extracted in full. To this end, we could integrate the measures describing the usage patterns of linking words. In this way we could calculate a score to help automate the decision-making process of including or not the information provided through these words. This mechanism would be a flexible way to determine the most appropriate limits for the formal unit of the explanation, where we want to maximize the amount of information, while having a sequence of terms as short as possible.

To reduce the false positive cases, it would be very useful to implement a module that uses the hyperonym relations extracted from RoWordNet (www.racai.ro/tools/text/rowordnet) to determine valid hyponym-hyperonym pairs. In this case, the role of the hyponym will be taken by the defined concept and thus we could determine a degree of confidence of the sentence by checking whether or not the super-concept is among the list of hyperonyms obtained from RoWordNet.

Another category of changes concerns the representation language of simplification patterns, to which a number of improvements could be made to increase its flexibility. One of these would be the possibility to define sequences of repeatable components, because at this point, only one component can be repeatable. Also, the components having indefinite type are currently allowed only for annotation testing and cannot have modifiers, as they would become much too general. A possible remedy for this issue is the introduction of modifiers with a limited number of applications, and here we are primarily referring to the recurrence modifier.

A major limitation is the lack of an adequate mechanism to ensure data persistence. At this point before each use it is necessary to perform a prior training phase. This would also facilitate the manual editing of patterns using the language defined for the representation of simplification patterns, which would lead to a positive increase in their generality. The problem that needs to be solved in this case is the recognition of these manual changes and their integration into the pattern, given that for them, there will be no additional lexico-syntactic information that is available for automatically created pattern components. Here, we face two situations: adding additional components to the pattern and modifying existing ones.

8 List of published and presented works

Niculiţă, C. and Dumitriu, L., 2020, October, An Experiment on Text Summarization: Frequent Terms and Concept Definition Extraction (accepted). *In 2020 24th International Conference on System Theory, Control and Computing, Sinaia, Romania*

Niculiţă, C. and Dumitriu, L., 2019, October. The Relational Parts of Speech in Text Analysis for Definition Detection, for Romanian Language. *In 2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet)* (pp. 135-140). IEEE.

Șușnea, I., Panait, O., **Niculiță, C.** and Munteanu, D., 2018. Machine Vision for Autonomous Vehicles-Potential and Limitations. A Literature Review. *The Annals of "Dunărea de Jos" University of Galati. Fascicle III, Electrotechnics, Electronics, Automatic Control, Informatics*, *41*(2), pp.24-30.

Niculiță, C., 2018. Survey on knowledge representation approaches for natural language processing. *The Annals of "Dunărea de Jos" University of Galati. Fascicle III, Electrotechnics, Electronics, Automatic Control, Informatics*, *41*(1), pp.5-12.

Niculiță C., Istrate A., Vlase M., Jâșcanu N., 2004. The Business Level Structure of WeBLE Platform. Load Tests. *Proceedings of The 12th International Symposium on Modeling, Simulation and Systems' Identification – SIMSIS 12*, Galați, Romania, ISBN 973-627-156-0

Bibliography

[1] Poon, H. and Domingos, P., 2010, July. Unsupervised ontology induction from text. In *Proceedings of the 48th annual meeting of the Association for Computational Linguistics* (pp. 296-305).

[2] Borg, C., Rosner, M. and Pace, G., 2009, September. Evolutionary algorithms for definition extraction. In *Proceedings of the 1st Workshop on Definition Extraction* (pp. 26-32).

[3] Veyseh, A.P.B., Dernoncourt, F., Dou, D. and Nguyen, T.H., 2020. A Joint Model for Definition Extraction with Syntactic Connection and Semantic Consistency. In *AAAI* (pp. 9098-9105).

[4] Navigli, R. and Velardi, P., 2010, July. Learning word-class lattices for definition and hyperonym extraction. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (pp. 1318-1327).

[5] Storrer, A. and Wellinghoff, S., 2006, May. Automated detection and annotation of term definitions in German text corpora. In *LREC* (pp. 2373-2376).

[6] Malaisé, V., Zweigenbaum, P. and Bachimont, B., 2004. Detecting semantic relations between terms in definitions. In *Proceedings of CompuTerm 2004: 3rd International Workshop on Computational Terminology* (pp. 55-62).

[7] Trimble, L., 1985. *English for science and technology: A discourse approach*. Cambridge University Press.

[8] Flowerdew, J., 1992. Definitions in science lectures. *Applied linguistics*, *13*(2), pp.202-221.

[9] Del Gaudio, R., Batista, G. and Branco, A., 2014. Coping with highly imbalanced datasets: A case study with definition extraction in a multilingual setting. *Natural Language Engineering*, *20*(3), pp.327-359.

[10] Meyer, I., 2001. Extracting knowledge-rich contexts for terminography. *Recent advances in computational terminology*, *2*, p.279.

[11] Auger, A., 1997. *Repérage des énoncés d'intérêt définitoire dans les bases de données textuelles* (Doctoral dissertation, Université de Neuchatel).

[12] Pearson, J., 1998. *Terms in context* (Vol. 1). John Benjamins Publishing.

[13] Alarcón, R., Sierra, G. and Bach, C., 2007. Developing a Definitional Knowledge Extraction System. In *Conference Proceedings of Third Language & Technology Conference LTC'07*.

[14] Reiplinger, M., Schäfer, U. and Wolska, M., 2012, July. Extracting glossary sentences from scholarly articles: A comparative evaluation of pattern bootstrapping and deep analysis. In *Proceedings of the ACL-2012 special workshop on rediscovering 50 years of discoveries* (pp. 55-65).

[15] Durán Muñoz, I., 2010. Specialized lexicographical resources: a survey of translators' needs. *Granger, Sylviane & Magali Paquot (eds.)*, pp.55-66.

[16] Weiten, W., Deguara, D., Rehmke, E. and Sewell, L., 1999. University, community college, and high school students' evaluations of textbook pedagogical aids. *Teaching of psychology*, *26*(1), pp.19-21.

[17] Cui, H., Kan, M.Y. and Chua, T.S., 2007. Soft pattern matching models for definitional question answering. *ACM Transactions on Information Systems (TOIS)*, *25*(2), pp.8-es.

[18] Duan, W. and Yates, A., 2010, June. Extracting glosses to disambiguate word senses. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 627-635).

[19] Navigli, R., Velardi, P. and Faralli, S., 2011, June. A graph-based algorithm for inducing lexical taxonomies from scratch. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

[20] Pantel, P. and Ravichandran, D., 2004. Automatically labeling semantic classes. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004* (pp. 321-328).

[21] Yang, H. and Callan, J., 2009, August. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP* (pp. 271-279).

[22] Kozareva, Z. and Hovy, E., 2010, October. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 1110-1118).

[23] De Benedictis, F., Faralli, S. and Navigli, R., 2013, August. Glossboot: Bootstrapping multilingual domain glossaries from the web. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 528-538).

[24] Tuan, L.A., Kim, J.J. and Ng, S.K., 2014, October. Taxonomy construction using syntactic contextual evidence. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 810-819).

[25] Seitner, J., Bizer, C., Eckert, K., Faralli, S., Meusel, R., Paulheim, H. and Ponzetto, S.P., 2016, May. A Large DataBase of Hyperonymy Relations Extracted from the Web. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 360-367).

[26] Klavans, J.L. and Muresan, S., 2001. Evaluation of the DEFINDER system for fully automatic glossary construction. In *Proceedings of the AMIA Symposium* (p. 324). American Medical Informatics Association.

[27] Espinosa-Anke, L., Saggion, H. and Ronzano, F., 2015, June. Taln-upf: Taxonomy learning exploiting crf-based hyperonym extraction on encyclopedic definitions. In

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015) (pp. 949-954).

[28] Degórski, L., Marcinczuk, M. and Przepiórkowski, A., 2008, May. Definition Extraction Using a Sequential Combination of Baseline Grammars and Machine Learning Classifiers. In *LREC*.

[29] Ide, N., Bonhomme, P. and Romary, L., 2000. XCES: An XML-based encoding standard for linguistic corpora. In *Proceedings of the Second International Language Resources and Evaluation Conference. Paris: European Language Resources Association*.

[30] Schmid, H., 2013, November. Probabilistic part-ofispeech tagging using decision trees. In *New methods in language processing* (p. 154).

[31] Jurafsky, D., 2000. Speech & language processing. Pearson Education India.

[32] Brants, Thorsten., 2000. TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied natural language processing*. Association for Computational Linguistics.

[33] Faralli, S. and Navigli, R., 2013, August. A java framework for multilingual definition and hyperonym extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (pp. 103-108).

[34] Westerhout, E. and Monachesi, P., 2007. Extraction of Dutch definitory contexts for elearning purposes. *LOT Occasional Series*, *7*, pp.219-234.

[35] Toutanova, K. and Manning, C., 2000. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the 2000 Joint SIGDAT Conference EMNLP/VLC, 63-71, 2000*.

[36] Branco, A. and Silva, J., 2006. A suite of shallow processing tools for portuguese: Lx-suite. In *Demonstrations*.

[37] Espinosa-Anke, L., Ronzano, F. and Saggion, H., 2015. Weakly supervised definition extraction. In *Angelova G, Bontcheva K, Mitkov R, editors. International Conference on Recent Advances in Natural Language Processing 2015 (RANLP 2015); 2015 Sept 7-9; Hissar, Bulgaria. Stroudsburg: ACL (Association for Computational Linguistics); 2015. p. 176-85.*. ACL (Association for Computational Linguistics).

[38] Spala, S., Miller, N.A., Yang, Y., Dernoncourt, F. and Dockhorn, C., 2019, August. DEFT: A corpus for definition extraction in free-and semi-structured text. In *Proceedings of the 13th Linguistic Annotation Workshop* (pp. 124-131).

[39] Boella, G., Di Caro, L., Ruggeri, A. and Robaldo, L., 2014. Learning from syntax generalizations for automatic semantic annotation. *Journal of Intelligent Information Systems*, *43*(2), pp.231-246.

[40] Esteche, J., Romero, R., Chiruzzo, L. and Rosá, A., 2017. Automatic Definition Extraction and Crossword Generation From Spanish News Text. *CLEI ELECTRONIC JOURNAL*, *20*(2).

[41] Ravichandran, D. and Hovy, E., 2002, July. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual meeting of the association for Computational Linguistics* (pp. 41-47).

[42] Saggion, H., 2004, May. Identifying Definitions in Text Collections for Question Answering. In *LREC*.

[43] Montes-y-Gómez, M., Pineda, L.V., Pérez-Coutiño, M.A., Soriano, J.M.G., Arnal, E.S. and Rosso, P., 2005, September. INAOE-UPV Joint Participation in CLEF 2005: Experiments in Monolingual Question Answering. In *CLEF (Working Notes)*.

[44] Westerhout, E., 2009, September. Definition extraction using linguistic and structural features. In *Proceedings of the 1st Workshop on Definition Extraction* (pp. 61-67).

[45] Cafarella, M.J., Downey, D., Soderland, S. and Etzioni, O., 2005, October. Knowitnow: Fast, scalable information extraction from the web. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing* (pp. 563-570).

[46] Velardi, P., Faralli, S. and Navigli, R., 2013. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, *39*(3), pp.665-707.

[47] Fahmi, I. and Bouma, G., 2006. Learning to identify definitions using syntactic features. In *Proceedings of the Workshop on Learning Structured Information in Natural Language Applications*.

[48] Monachesi, P. and Westerhout, E., 2008. What can NLP techniques do for eLearning. *INFOS2008 proceedings*, pp.150-156.

[49] Espinosa-Anke, L. and Schockaert, S., 2018, June. Syntactically aware neural architectures for definition extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)* (pp. 378-385).

[50] Chen, C., Liaw, A. and Breiman, L., 2004. Using random forest to learn imbalanced data. *University of California, Berkeley*, *110*(1-12), p.24.

[51] Schroeder, J., Cohn, T. and Koehn, P., 2009, March. Word lattices for multi-source translation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)* (pp. 719-727).

[52] Pustet, R., 2003. *Copulas: Universals in the Categorization of the Lexicon*. OUP Oxford.

[53] Marcus, M., Santorini, B. and Marcinkiewicz, M.A., 1993. Building a large annotated corpus of English: The Penn Treebank.

[54] Manning, C.D. and Schütze, H., 2002. *Foundations of Statistical Natural Language Processing*. 5th ed., MIT Press, Cambridge, MA.

[55] Manning, C.D., 2011, February. Part-of-speech tagging from 97% to 100%: is it time for some linguistics?. In *International conference on intelligent text processing and computational linguistics* (pp. 171-189). Springer, Berlin, Heidelberg.

[56] Tufiş, D., Barbu, A.M., Pătraşcu, V., Rotariu, G. and Popescu, C., 1997. Corpora and corpus-based morpho-lexical processing. *Recent Advances in Romanian Language Technology, Editura Academiei*, pp.35-56.

[57] Simionescu, R., 2011. Hybrid pos tagger. In *Proceedings of Language Resources and Tools with Industrial Applications Workshop (Eurolan 2011 Summer School), Cluj-Napoca, Romania* (pp. 21-28).

[58] Simionescu, R., 2012. Romanian deep noun phrase chunking using graphical grammar studio. In *Proceedings of the 8th International Conference "Linguistic Resources And Tools For Processing Of The Romanian Language* (pp. 135-143).

[59] Zafiu, A., Dumitrescu, S.D. and Boroş, T., 2015. Modular language processing framework for lightweight applications (MLPLA). In *7th Language & Technology Conference*.

[60] Straka, M., Hajic, J. and Straková, J., 2016, May. UDPipe: trainable pipeline for processing CoNLL-U files performing tokenization, morphological analysis, pos tagging and parsing. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16) (pp. 4290-4297).

[61] Tufis, D., 2000, May. Using a Large Set of EAGLES-compliant Morpho-syntactic Descriptors as a Tagset for Probabilistic Tagging. In *LREC*.

[62] Science Encyclopedia, Litera Ed., 2018, 304 pg., Bucharest, ISBN 978-606-33-3101-5

Appendix I

```
public enum AnnotationType {
    LINKWORD("LINK"),
    PARTICIPLE_FORM("PART"),
    PARTICIPLE_FORM_COLLAPSED("PARTCOL"),
    GERUND_FORM("GER"),
    PREC_ATTRIBUTE("PATTR"),
    ADVERB_BREAK("ADVBR"),
    NPLIMIT("NLIM"),
    CONJUNCTION("CNJBR"),
    CONJUNCTION_ANDOR("CNJAO"),
    ADPOSITION("ADPBR"),
    PUNCTUATION("PUNCT"),
    NP_UNDEF("NPUNDEF"),
    NP_DEF("NPDEF"),
    NP_IGNORE("NP_IGN"),
    NP_TARGET("NP_TARG"),
    MULTIPLE_WORD_NOUN("MULTNOUN"),
    OVERRIDE("OVRD"),
    ONEOF("ONEOF"),
    COMPGRADE("CMPGR"),
    ADJ_LIKE("ADJ_LIKE"),
    ENUMERATION("ENUM"),
    OBLIQUE("OBLQ"),
    LW_CAREO("LW_CAREO"),
    LW_ALCAREO("LW_ALCAREO"),
    PP("PP");
}
```

Listing A-1 Predefined annotation labels

Appendix II

```
    //enum section
    private static final double ST_COMMA = 0;
    private static final double ST_AND_OR = 50;
    private static final double ST_MULTIPLE_CONJ = -100;
    private static final double ST_BOTH_AND_OR = -2000;
    private static final double ST_BOTH_COMMA_CONJ = -30;
    private static final double ST_SINGLE_NOUN_WITH_ENUM = -500;

    private static final double ST_UNDEFINED = -10000;


    //enum only
    //private static final double PREV_BONUS = 25;
    private static final double ST_COMMA_CONJ_HANDICAP = -25;
    private static final double ST_SIMILAR_HANDICAP_MULTIPL = -60;
    //noun only
    private static final double ST_ADJ_NOUN_MIX = -50;

    //noun section
    private static final int ST_IDENTICAL = 100;
    private static final int ST_VERY_SIMILAR = 80;
    private static final int ST_RGET_VERY_SIMILAR = 90;
    private static final int ST_SOMEWHAT_SIMILAR = 50;
    private static final int ST_TOTALLY_DIFFERENT = 0;
    private static final int ST_UNKNOWN = -1;

    private static final double ST_COMMON_PROPER_FRACTION = 0.8;

    private static final int ST_LW_MAX_SIMILARITY = 50;

    private static final int ST_MAX_SIMILARITY = 150;

    private static final int ST_NOUNTYP_COMMON = 35;
    private static final int ST_NOUNTYP_PROPER = 85;

    private static final boolean ST_GROUP_COMMON_AND_PROPER_NOUNS = true;
    private static final boolean ST_RGET_GROUP_COMMON_AND_PROPER_NOUNS = false;

    //used only for common nouns
    //the type order is DEFINED (0), UNDEFINED (1), NONE(2), UNKNOWN(3) article
on both columns and lines
    private static final int[][] artTypePairs_st = {
            {50, 40, 15, 0},
            {40, 50, 45, 0},
            {20, 30, 50, 0},
            {0, 0, 0, 0}
    };

    private static final int[][] artTypePairs_st_rget = {
            {50, 0, 0, 0},
            {0, 50, 45, 0},
            {0, 0, 50, 0},
            {0, 0, 0, 0}
    };

    //the case order is N_AC(0), G_D(1), UNKNOWN(2)
    //UNKNOWN type should not appear in practice
    private static final int[][] nounCasePairs_st = {
            {50, 0, 0},
            {0, 50, 0},
            {0, 0, 0}
    };

    private static final int ST_LW_SAME_LINKWORD = 10;

    private static final int LWM = ST_LW_MAX_SIMILARITY;

    //the type order is STRONG(0), MEDIUM(1), WEAK(2), STOP(3), NONE(4),
UNKNOWN(5)
    private static final int[][] lwTypePairs_st = {
            {LWM, 0, 15, 0, LWM, 0},
            {0, LWM, 0, 0, LWM, 0},
            {15, 0, LWM, 0, LWM, 0},
            {0, 0, 0, LWM, LWM, 0},
            {15, 15, 15, 15, LWM, 0},
            {0, 0, 0, 0, 0, 0}
    };

    private static final int[][] lwTypePairs_st_rget = {
            {LWM, 0, 15, 0, LWM, 0},
            {0, LWM, 0, 0, LWM, 0},
            {15, 0, LWM, 0, LWM, 0},
            {0, 0, 0, LWM, LWM, 0},
            {0, 0, 0, 0, LWM, 0},
            {0, 0, 0, 0, 0, 0}
    };
```

Listing A-2 Numerical constants used in enumeration detection