PROCESSING TURKISH RADIOLOGY REPORTS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KEREM HADIMLI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

MAY 2011

Approval of the thesis:

PROCESSING TURKISH RADIOLOGY REPORTS

submitted by **KEREM HADIMLI** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen
Dean, Graduate School of **Natural and Applied Sciences** ________________

Prof. Dr. Adnan Yazıcı
Head of Department, **Computer Engineering** ________________

Prof. Dr. Göktürk Üçoluk
Supervisor, **Computer Engineering Department, METU** ________________

Dr. Meltem Turhan Yöndem
Co-supervisor, **Faculty of Eng. and Natural Sciences, Sabancı U.** ________________

Examining Committee Members:

Dr. Ayşenur Birtürk
Computer Engineering Dept., METU ________________

Prof. Dr. Göktürk Üçoluk
Computer Engineering Dept., METU ________________

Dr. Onur Tolga Şehitoğlu
Computer Engineering Dept., METU ________________

Dr. Cevat Şener
Computer Engineering Dept., METU ________________

Dr. Meltem Turhan Yöndem
Faculty of Engineering and Natural Sciences, Sabancı University ________________

Date: ________________

ABSTRACT

PROCESSING TURKISH RADIOLOGY REPORTS

Hadımlı, Kerem
M.Sc., Department of Computer Engineering
Supervisor : Prof. Dr. Göktürk Üçoluk
Co-Supervisor : Dr. Meltem Turhan Yöndem

May 2011, 86 pages

Radiology departments utilize various visualization techniques of patients' bodies, and narrative free text reports describing the findings in these visualizations are written by medical doctors. The information within these narrative reports is required to be extracted for medical information systems. Turkish is an highly agglutinative language and this poses problems in information retrieval and extraction from Turkish free texts.

In this thesis one rule-based and one data-driven alternate methods for information retrieval and structured information extraction from Turkish radiology reports are presented. Contrary to previous studies in medical NLP systems, both of these methods do not utilize any medical lexicon or ontology.

Information extraction is performed on the level of extracting medically related phrases from the sentence. The aim is to measure baseline performance Turkish language can provide for medical information extraction and retrieval, in isolation of other factors.

Keywords: natural language processing, information retrieval, information extraction, rule based algorithm, data driven algorithm

ÖZ

TÜRKÇE RADYOLOJİ RAPORLARININ İŞLENMESİ

Hadımlı, Kerem
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Göktürk Üçoluk
Ortak Tez Yöneticisi : Dr. Meltem Turhan Yöndem

Mayıs 2011, 86 sayfa

Radyoloji bölümleri hastaların vücutlarını görselleştiren teknikler kullanır ve bu görüntüler doktorlar tarafından incelenerek düz metin raporlara dökülür. Medikal bilgi sistemleri açısından bu düz metin raporlardaki bilginin çıkartılması önem arz eder. Türkçe'nin oldukça eklemeli bir yapıya sahip oluşu bilgi erişimi ("information retrieval") ve bilgi çıkarımı ("information extraction") için çeşitli zorluklar oluşturmaktadır.

Bu tezde biri kural tabanlı biri veri tabanlı olmak üzere Türkçe radyoloji raporlarının işlenmesinde kullanılabilecek iki yöntem önerilmektedir. Önceki medikal doğal dil işleme çalışmalarının aksine, her iki yöntemde de bir medikal sözlük ya da bir medikal ontoloji kullanılmamaktadır.

Bilgi çıkarımı, verilen cümle içinde medikal olarak ilişkili sözcük öbeklerinin ve bunların ilişkilerinin belirlenmesi seviyesinde yapılmaktadır. Amaç, diğer etkenlerin yokluğunda, Türkçe'nin özelliklerinin medikal bilgi çıkarımı ve erişimi için sunabileceği referans performansı belirlemektir.

Anahtar Kelimeler: doğal dil işleme, bilgi erişimi, bilgi çıkarımı, kural tabanlı algoritma, veri tabanlı algoritma

ACKNOWLEDGMENTS

This study was supported by Turkish Scientific and Technological Research Council (TÜBİTAK) project 3080179. I would like to thank to Radiology Department of Hacettepe University Hospital; Prof. Dr. Deniz Akata and Prof. Dr. Mustafa Özmen for their support.

I would like to thank to Dr. Meltem Turhan Yöndem for her encouragement and invaluable support.

I am grateful to all permanent and semi-permanent residents of A-206; Okan Akalın, Can Hoşgör, Utku Erdoğdu, Elvan Gülen, Merve Aydınlılar.

I also thank to Selma Süloğlu for all her good intentions and support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Radiology departments utilize different 2-dimensional and 3-dimensional medical imaging technologies to create visualizations of patients' bodies. These 2-D and 3-D images are systematically analyzed by doctors and free text narrative reports describing their findings in the images are written. These reports contain a vast amount of medical information and assume shared knowledge between writer and reader [38].

Extracting structured information from these free text reports requires knowledge on the field and the language. These reports contain a vast amount of medical information, and the reader is assumed to have sufficient medical background. In spite of this assumption, it has come to our attention that a person with no medical knowledge can still understand the basic relations within Turkish radiology report sentences. Even in the lack of medical knowledge, these relations may be used to deduce if a related phrase shows an anatomical location, a finding, or a quality; although some of these deductions would be incorrect due to missing medical education and knowledge.

This thesis is part of a larger project on information retrieval from Turkish radiology reports, supported by TÜBİTAK and Hacettepe Hospital Radiology Department. The project consists of three different but related parts. Output of these three parts are combined to cover shortcomings of each single part. First part of the project is a tagging tool. This tool provides doctors an option to associate continuous phrases in text with SNOMED-CT concept identifiers. These associations are later used to tag new reports automatically. This part can only tag new reports if they contain exactly the same Turkish phrases. The second part of the project is a translation and ontology search engine. It makes uses of a Turkish-English medical dictionary, previously tagged Turkish phrases, their SNOMED-CT identifiers, and ontological relations within SNOMED-CT, in order to find SNOMED-CT identifiers of new phrases in text (that were never tagged before). The third part is formed by this thesis, to provide a translation-free and ontology-free framework for information retrieval and extraction in context of radiology reports.

Turkish is characterized by a rich agglutinative morphology, free constituent order, and predominantly head-final dependencies [11]. Word structures in Turkish are formed by productive affixations of derivational and inflectional suffixes to root words. Morphemes appended to a word can change the word's part of sentence tag, and these morphemes may be appended consecutively to form a series of different meanings [26]. The most important difficulty in analyzing Turkish free text lies in the number of possible part of speech (POS) tags that can be assigned to a single word. Syntactic relations within a sentence are partly determined by morphological features, and the number of morphemes that can be appended to a word in Turkish is theoretically unlimited. Nearly half of the words in a sentence are morphologically ambiguous. Moreover, only the final POS tag of words are not sufficient to completely analyze a sentence, as syntactic relations may be formed between a word's earlier inflectional groups and other words in the sentence [41, 27, 29].

Information extraction and retrieval from Turkish radiology reports is more challenging than English counterparts. Besides the general difficulties related to Turkish which are mentioned, radiology reports present more challanges as there is no consensus on Turkish spelling of most medical terms derived from western languages. Each radiologist tends to write the same terms differently. This is not a misspelling problem as is common in NLP context, but rather a choice caused by differences in medical education.

Another problem lies in the agglutinative nature of Turkish. Not only morphological analyses of common words, but also morphological analyses of medical terms are required in order to analyze Turkish sentences. This requirement poses a problem with medical terminology in terms of correctly identifying and disambiguating alternative morphological analyses.

In this thesis two different methods for structured information extraction from Turkish radiology reports are proposed. Both methods are shaped around the idea of using only Turkish grammar and no advance information on medical terminology or relations. The aim is to identify the baseline performance Turkish language might provide for this task.

The contents of this thesis are organised as follows:

- In Chapter 2, a survey on related work and previous medical natural language processing systems is given.
- In Chapter 3, detailed background information on the base assumptions and methods are provided, which provide a basis for the following chapters.
- Chapter 4 contains details of the proposed methods. It is further subdivided into rule-based and data-driven methods.
- Chapter 5 provides information on datasets and metrics used for evaluation. Rule-based method is evaluated for document retrieval performance. Both rule-based and data-driven methods are evaluated for extraction performances. Evaluation results are analyzed and discussed.
- Chapter 6 concludes the thesis and supplies ideas on how the results of this work can be utilized in future work.

CHAPTER 2

LITERATURE SURVEY

Although English is the most common target, medical natural language processing systems are already being used for various other languages. Spyns [37] gives an overview of medical NLP systems prior to 1995. The most important objective is information retrieval, whereas indexing and encoding are derived applications. These systems cover medical text in English, Dutch, French, German and Japanese, and some of them are multilingual. Most of these systems use Conceptual Graph formalism for representing information. Another preference is use of “frames”. A “frame” holds all related properties of a single clinical finding within a report. Friedman [15] provides a view on state of the art systems by 1999. Medical ontologies and dictionaries are widely used. SNOMED-CT, ICD-10, UMLS (Unified Medical Language System) and SPECIALIST Lexicon are available.

SNOMED-CT is a comprehensive medical terminology, designed for expressing clinical documentation and reporting. It contains terminology on clinical findings, procedures, body structures, external physical objects, physical forces, and various other entities that could be encountered in clinical documentation. SNOMED-CT establishes medically relevant relations between the concepts it represents, such as *IS-A* relations.

ICD-10 (International Statistical Classification of Diseases and Related Health Problems, 10^{th} Revision) is a diagnostic classification standard that came into use by WHO (World Health Organization) member countries beginning in 1994. ICD-10 is an international standard for health management and clinical use.

UMLS (Unified Medical Language System) is a collection of medical concepts and related tags from a variety of tagging systems, such as SNOMED-CT and ICD-9. UMLS’s primary purpose is to map concept identifiers from different tagging system together, enabling information exchange between different clinical environments.

SPECIALIST Lexicon is one of the knowledge sources of UMLS, initially developed to provide lexical information for SPECIALIST Natural Language Processing System. It is designed for use with English, and covers many biomedical terms, as well as English vocabulary.

Use of ontologies in English for processing medical reports in other languages may not be feasible due to translation and matching problems. In these cases, usually a small ontology and lexicon is constructed manually. Mykowiecka et al. [21] use a rule based method with a specially constructed ontology to extract information from Polish mammography reports. Another possibility is the use of translation approaches. Castilla et al. [3] use a machine translation framework augmented by medical vocabulary to translate Polish chest radiology reports into English. Once translated into English, the reports can be processed with proven Medical Language Processing (MLP) systems like MEDLEE [13].

Instead of indexing text, some methods try to keep the extracted information pinned to the original text for future access. An example is automatically enhancing free text reports with automatically generated XML [16].

Many information extraction systems are specially designed for a single domain. Domain experts have to create rules or extensively annotate a training set. Due to the inherent domain dependence, porting existing systems to other domains require a vast amount of work, and most of the time not feasible. Alternatively, adaptive information extraction systems exist which are domain independent. Instead of using works of domain experts, adaptive information extraction systems utilize Machine Learning approaches, and only rely on simple training sets. These training sets are annotated only for the target results. [6, 19, 40]

Evaluating a system with ineffective evaluation methods weakens any good performance measures obtained. But evaluating NLP systems in medical domain is a difficult task.

Friedman and Hripcsak [14] group proper evaluation criteria into five groups. Minimizing bias; so that developers don't see the test set. Establishing a reference standard; so that the test set is constructed by the domain experts, is large enough, and covers the goals of the study. Describing evaluation methods; presenting results; and discussing conclusions; so that the reliability of the evaluation method can be double checked by the audience. They note that in medical domain, experts may be biased or inconsistent, even with their own previous decisions on interpreting patient records.

In order to measure the reliability of experts that create the reference standard, reliability studies are performed. Experts may rate others so that distance among experts can be measured, or an error rate can be calculated among experts. In examining two studies focused on radiology reports, Hripscak et al. [18] state that only a single rater is reliable enough to create a reference standard and to measure the overall performance of a system. But, they also state that if case-by-case assessments are required, six raters are necessary.

A problem with data driven methods lies in creation of training set. In NLP context, annotating a training set properly is a very time consuming task. Redundantly annotating similar examples add little value to the resulting training set. A method to reduce annotation cost is sample selection. In this method, while learning, the system decides which next sample will provide the most information gain, and asks the operator to annotate it. Using a single model to measure uncertainty in candidate samples is an option, but this would not disambiguate between (a) uncertainty due to insufficient training examples and (b) uncertainty due to inherent ambiguity. Engelson and Dagan [7] favor committee-based selection methods, so that a candidate sample can be compared with multiple models, and become the next sample only if multiple models disagree.

In the case of processing Turkish radiology reports, the first known work is by Soysal [34, 35]. The implemented system utilizes manually coded rules to create a mapping from Turkish free-text reports to properties of SNOMED-CT identifiers (propertries such as measurement, existence, abnormality, etc.). The prototype system is designed only for abdominal ultrasonography reports, and makes use of a manually created ontology besides the rules.

CHAPTER 3

BACKGROUND

This chapter provides information on background topics that provide a basis for this thesis. These topics are morphological analysis, morphological disambiguation, dependency grammar (and parsing), and conceptual graphs.

Morphological analysis is used for retrieving probable analyses (roots and morphemes) of input word strings. Morphological disambiguation is required as a single input word might have multiple ambiguous analyses. Dependency parsing is used for syntactically analyzing input sentences. Conceptual graphs are utilized for knowledge representation purposes.

Additionally, information on tools that are used related to these topics are provided in the last section of this chapter.

3.1 Morphological Analysis

Turkish is an agglutinative language with productive inflectional and derivational processes. Morphemes attached to a word may cause inflection or derivation. Derivations bring a different but related word to existence, possibly changing the part of speech category of the word. In Turkish Treebank [29], derivational boundaries (DBs) are used to separate inflectional groups (IGs). An inflectional group consists of a single POS tag and some inflectional features. Separation of inflectional groups (the derivation boundaries) are determined by derivational morphemes.

Inflection

Inflection is the surface form change of a lexeme by attaching a morpheme. The lexeme retains its meaning, but changes in the way it interacts with other words in the sentence. An example is the locative morpheme.

Derivation

Word formation may happen by means of derivation or compounding. In word formation, a related but different lexeme comes into existance. The new lexeme's grammatical category (POS) is usually different than the old lexeme. Derivational morphemes are used extensively in Turkish language, where most of the lexemes used are formed by derivation of others.

Grammatical Category (Part of Speech)

Each lexeme can be grouped into a single grammatical category, or part of speech tag. In Oflazer's morphological analyzer implementation, nouns, adjectives, verbs, compound nouns, proper nouns, pronouns, adverbs, connectives, exclamations, postpositions, acronyms are used. The choice of POS grouping may depend on the analyzer.

Lexemes can have different morphotactics based on their grammatical category, but don't have to. For example adjectives and nouns share the same morphotactics in Turkish language, but their POS difference affects sentence structure. Grammatical category first affects which morphemes the lexeme will accept. Also it affects the analysis of the structure of a sentence. Intra-sentence relations of words are defined by lexemes' grammatical categories and inflections. Change of grammatical category is possible in Turkish by derivational morphemes.

Two-level morphology: Overt, Covert and Null Morphemes

Not all morphemes have to affect the surface realization of a lexeme. Also, not every surface realization has to correspond to a single (or any) morpheme. If a morpheme exists in a lexeme and affects its surface form, the morpheme is said to be overt. If a morpheme's existence is defined by non-existance of some other morphemes, the morpheme is said to be covert.

A good example is the plural suffix *-s* in English. In the words *dogs* and *cats*, the plural morpheme is present and has a surface form. If the overt plural morpheme is not present, *dog* and *cat* are said to have a singular morpheme (covert) attached, although no surface realization can be seen.

Null morphemes also do not affect a lexeme's surface form, but are different than overt morphemes. A null morpheme's existance is not denoted by non-existence of other morphemes, but by the context the word appears in. An example is the word *sheep*, which may have singular or plural null morphemes attached depending on the context.

Two-level morphology is a paradigm for morphological description of word structures [20, 26, 36]. It is used for analyzing word structures in different languages. Two-level morphology consists of surface form rules and lexical form rules. Both rule sets are applied in parallel in order to analyze or generate a word. The rules are usually constructed and applied within a finite state transducer framework.

Tagsets, ecosystems, interoperability

Although there are currently multiple morphological analyzers available for Turkish, there is no single morpheme tagset that is accepted universally. Thus, every different morphological analyzer devises its own notations, some with minor differences to existing systems, and some with complete rewrites.

Higher levels of NLP toolsets (statistical/dependency parsers, morphological disambiguators) rely on morphological analysis outputs. In data driven studies that cover Turkish language generally, even if training set is large enough to generate useful models for future studies, the generated models may not be usable if they are trained for a different analyzer than expected. Grouping studies into non-interoperable islands, this poses an important problem for Turkish NLP studies.

Oflazer [26] provided the first two-level morphological analyzer for Turkish. Eryiğit [8, 22] provided an affix stripping analyzer, which is unique in that it does not need any lexicon to analyze words. Zemberek project [1] is a free-software surface-form oriented morphological analyzer, which filled an important gap. TRMorph [4] is a freely available two-level morphological analyzer constructed on Stuttgart Finite State Transducer framework, and is also free-software.

3.2 Morphological Disambiguation

In a highly agglutinative language like Turkish, morphemes play a very important role for the syntactic relations between words. The problem lies in that morphological analyses of a typical word is highly ambiguous in Turkish, and there is usually more than one analysis for a single word. In their test with a corpora consisting of 950,000 words, Yuret and Türe [41] note that there is on average 1.76 alternative parses for any word.

Morphological disambiguation is more complex than assigning the correct POS tag to each word, as other morphemes (both inflectional and derivational) would also affect the syntactic relations [11]. The number of potential combinations of morphemes is unlimited, and the numbers are still high even when calculated per inflectional group. Morphological analyzers used in previous studies cover nearly 120 different morphemes, which can be appended to words in a circular manner, producing unlimited number of inflectional groups and affixes. The possible number of combinations per inflectional group is rather limited, but still on the order of thousands [17].

Previously proposed disambiguators for Turkish use heuristics [28], purely statistical approaches [17], and rule learning per different morpheme [41]. For statistical disambiguators, data sparseness is an important problem.

3.3 Dependency Grammar and Parsing

First introduced by Tesnière [39], dependency grammar is a large and diverse family of grammatical theories and formalisms, all sharing the basic assumption that structure of a sentence is based on lexical elements. These elements are connected by binary asymmetric relations. Dependency grammar lacks the concept of phrasal nodes. A more detailed review of dependency grammar and different aspects of different theories can be found in Nivre's study [23].

Dependency grammar represents sentences as directed bilexical dependency links between lexemes. Compared to constituent based formalizations, dependency grammar is more adequate for free constituent order languages like Turkish [11].

In a sentence analysis, directed dependency links form between pairs of words, one being called the dependent (or modifier), and the other the head. This links are shown as directed arcs, but the arc's direction may be from head to dependent or vice-versa depending on the notation used.

Being an agglutinative language, Turkish poses interesting properties for parsing with dependency grammar. Words are extensively formed by derivations in Turkish. These derivations cause part of speech tags to change at derivational boundaries and each group has its own inflectional features. Although a word in a Turkish sentence has a final part of speech tag, its non-final inflectional groups (IGs) may have other POS tags. Inflectional groups are separated by derivational boundaries (DBs). Turkish Treebank [29] represents links between dependents and heads using these sublexical inflectional groups. Only the final inflectional group of a word can be a dependent, but any inflectional group can be a head. The reason is that only the last IG of a word determines the entire word's role as a dependent. Thus, dependency links emanate only from final inflectional groups, but land on any inflectional group of another word. Inside a single word, non-final inflectional groups may be shown as modifing the word's next inflectional group to complete the chain to the final group.

Data driven dependency parsing of Turkish is first researched by Eryiğit and Oflazer [10]. This statistical parser computes the probabilities of unit-to-unit dependencies and finds the most probable tree. In 2008, Eryiğit et al. [11] developed a classifier based parser for Turkish inside the classifier based parser framework MaltParser developed by the language technology group at Växjö University [24] and applied their work in CoNLL-X shared task [25]. Using of sublexical units called inflectional groups (IGs) as the main unit of dependency links is shared among these work (including the Turkish treebank). Although using whole lexemes were also analyzed, using IGs is shown to be the best performing unit [9].

3.4 Knowledge Representation and Conceptual Graphs

In an information extraction scenario, the value of knowledge representation cannot be underestimated. The format of the data representation identifies and limits the vocabulary and the capability of an information extraction system. Any information in the input text that is outside a system's representation capabilities will simply be lost.

In this work, use of a graph formalism is chosen for knowledge representation. Conceptual graphs provide an intuitive and easy to understand visualization ability to natural language processing systems.

Conceptual graphs were founded by Sowa's seminal work [32, 33]. They provided an intuitive way to represent the knowledge for natural language systems. During the years following, a community around conceptual graphs were found, whom contributed to the field. CGs are a family of formalism rooted in Sowa's work, rather than a single precise formalism [5].

The key difference between conceptual graphs (CGs) and other graph based formalisms is that, CGs are logically founded. CGs can be mapped to first order logic, which has been the foundation for knowledge representation and reasoning for millennia.

The conceptual graph notation that is used in this work is referred as a basic conceptual graph (BG). A BG is basically a labeled bipartite multigraph. Being a bipartite graph, its nodes can be grouped into two different kinds, and every edge in the graph only connects one node from each kind. Being a multigraph, there may be multiple different edges between two nodes. Every node and edge of the graph is labeled.

The two kinds of nodes are called relation nodes and concept nodes.

Relation nodes constitute the vocabulary (of relations) of the system. Each relation node has a label that denotes its type. Based on its type, a relation node has a predefined set of labeled edges it may be connected.

Concept nodes map to the entitites in the system. Although concept nodes are capable of representing bound and unbound variables, in this work only bound variables – specific entities will be used.

Figure 3.1 shows an example basic conceptual graph. Relation nodes are shown as rectangles, and concept nodes are shown as ovals. The naming of edges may have different notations, but in this work labeling them with strings is preferred. One of the other notations use numbers, and another notation uses directed edges (only in case all relation node types can have only two edges).

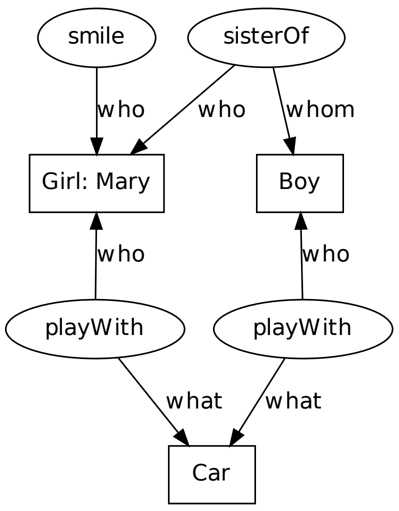


Figure 3.1: An example Basic Conceptual Graph (BG)

Rules

Rules map to implicit knowledge in a knowledge system. These rules can be used to deduce new information from existing. A rule represents an information like "if information H mexists in the knowledge, then information C can be deduced from it". H is called the hypothesis and C the conclusion.

The CG community defines a rule as a bicolored BG, so that vertices colored with white (and their edges) represent the hypothesis, and the vertices colored with gray (and their edges) represent the conclusion which can be added to the initial knowledge. Although a more general notation also exists, in this work bicolored graph notation will be used for CG rules.

An example rule is given in Figure 3.2. This rule states that if a person A has an uncle B, then this person A has a father (who is also a person), and this father has a brother who is person B.

3.5 Utilized Tools

Various external tools were used for the implementation and evaluation of this thesis. Table 3.1 provides information on which tools were used in which methods.

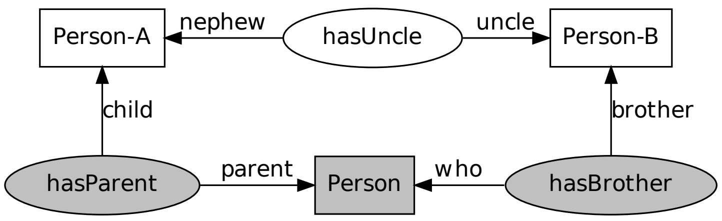


Figure 3.2: An example Conceptual Graph Rule

Table 3.1: Used tools vs. implemented methods

Tool	Purpose	Methods
Zemberek	Morphological analyzer	Rule-based
TRMorph	Morphological analyzer	Data-driven
Yuret and Türe, Disambiguator	Morphological disambiguator	Data-driven
MaltParser	Dependency parser	Data-driven
LIBLINEAR	Linear SVM Implementation	Data-driven

3.5.1 Zemberek

Zemberek [1] library is an NLP tool which is used for morphological analysis and spellchecking purposes in Turkic languages. It is used in many software packages for spellchecking. In this work Zemberek library is used to extract part of sentence tags and morphological tags of input words.

3.5.2 TRMorph

TRMorph [4] is a freely available two-level morphological analyzer for Turkish, which is based on Stuttgart Finite State Transducer tools. Being a two-level analyzer and based on a finite state transducer foundation, it is capable of providing more detailed analyses than Zemberek. Although other two-level morphological analyzers also exist for Turkish, currently TRMorph seems to be the only freely available one.

TRMorph is used in the data driven method, where a more detailed morphological analysis operation is required for dependency parsing.

3.5.3 Morphological Disambiguator

Yuret and Türe [41] implement a rule learner for disambiguation of Turkish morphemes, in which a different decision list is learned per morpheme. This approach successfully overcomes the data sparseness problem. The inputs to the decision lists are the all possible string suffixes of words within a neighborhood of 5 words. The decision lists are constructed by a greedy algoritm.

The decision lists act as oracles that tell whether a morpheme should or should not exist in a word (checking strings within 5 word window). Probabilities of the provided alternative morphological analyses can then be calculated (based on their aprior probabilities in training dataset and i.i.d. assumption), and the most probably analysis can be identified.

Yuret and Türe also point that the decision lists can be used as oracles by themselves, which turns the disambiguator into an analyzer. They achieve fairly good results for this experiment in their dataset.

3.5.4 MaltParser

MaltParser is a system for data-driven dependency parsing, developed by the language technology group at Växjö University [24]. Although the parser system has a few parsing alternatives, the parsing algorithm proposed by Eryiğit et al. [11] is used in this study.

The parser used is based on three components:

- A deterministic parsing algorithm for building dependency graphs
- An history-based model for predicting next action
- A classifier to map history to next parser action

The parser is entirely data-driven. The parsing algorithm is applied to sentences in training set, and a support vector machine classifier is trained for learning the action the parser needs to take using the already-tagged dependency graphs. In addition to probable dependency links, the classifier uses lexical forms (roots), part of speech categories and inflectional features of the tokens as inputs.

After the training period, the parser is able to work on new sentences. It never fails to produce an analysis for any input sentence due to the usage of a classifier.

3.5.5 LIBLINEAR

LIBLINEAR [12] library is used as a Linear Support Vector Machine classifier implementation. The library is mostly used in large scale classification of data, with millions of features and vectors, such as document classification.

A support vector machine is a general concept in which the training process finds the best hyperplane that separates labeled input vectors in two different regions in space. The hyperplane is guaranteed to be at the greatest distance from all input vectors. If there is no hyperplane completely separating the data vectors, an error penalty parameter and a threshold for maximum error guarantees maximum separation within the allowed error limits.

A kernel based SVM has the ability to map input vectors into an infinite dimensional space for separation. A linear SVM finds the best hyperplane in the input vectors' space.

Linear SVMs are used in this work due to computational time considerations.

CHAPTER 4

METHOD

In Section 4.1, the dataset used in this work is introduced. In Section 4.2 and Section 4.3, the methods used for information extraction are explained. Section 4.2 contains the details of the rule based method, which is also used in retrieval context. An alternative data driven method is proposed in Section 4.3. This data driven method is augmented with support vector machines in Section 4.3.5.

4.1 Data Description

The dataset used in this work is created from the example reports provided by Radiology Department of Hacettepe University Hospital. There are 4634 reports in total. These reports cover multiple radiology disciplines, and are written by different doctors. The reports are anonymous; patients' names and identifier numbers were already removed by the Radiology Department.

The reports are loosely structured into a few sections, and each section is free text. Each report contains a title and a list of doctors who wrote the report. The title field is almost from a finite set, sometimes with minor differences. A set of sample abdominal radiology reports are given in Appendix A.

There are five sections that contain text. Not all reports contain all of these sections. Also the names of the sections are sometimes inconsistent. Heuristics were required to extract the sections from reports.

Clinical information section contains the referring physician's notes on the reason of the request for radiologic imaging. *Technical information* or *Method* sections contain an explanation on preparation steps of the patient and how the imaging is performed. These sections do not appear in all reports.

In *Findings* section, the doctors explain what they see in the images. This section contains information on existing findings as well as non-existant ones. The order of the analyses of the images are nearly always determined per report type. The doctor goes through almost always the same steps (per report type) when examining the images, and writes everything she notices (of existance, nonexistance, normality and abnormality) into the report. This section contains full sentences.

A report ends with a *Conclusion* section. In this section the doctor picks the important findings from the *Findings* section, and paraphrases them. The conclusion section sometimes contains a listing of sentences where each item is about a different finding, sometimes a single sentence of findings. Contrary to the *Findings* section, *Conclusions* section usually contains sentences with no verbs, although this is not the case in all reports. If no abnormal findings were seen, *Conclusions* section contains a single sentence like "The results are normal" or "(Name of Imaging Method) within normal limits".

Findings and *Conclusions* sections are present in nearly all of the reports, although there are some outliers with no *Conclusions* or no *Findings* sections. The sections and context of the report depends on the doctor's preferences.

The statistics on the reports are given in Table 4.1. Statistics on report types are given in Table 4.2.

Table 4.1: Report statistics

Description	Total
Reports	4634
Unique types	287
Types w/ more than 10 reports	45

Conceptual graph formalism is chosen as output representation. Conceptual graph representation of the input sentences are constructed by both methods. This conceptual graph representation consists of relations like location, qualification, and quantification. Consecutive

Table 4.2: Distribution of example reports

Report Type (Coarse grouping)	Number of Reports
ABDOMINAL	840
THORACIC	944
HEAD, NECK, BRAIN, SPINAL	1942
JOINTS, BONES	252
OTHERS	656

words forming medically non-breakable phrases must be kept together as whole concepts in the output graph. The relations between concepts must be correct, with edges labeled properly showing the different roles of the affected concepts in a relation.

4.2 Rule-based Method

The first method uses only local templates and non-local rules on Turkish grammar. The templates are hand-crafted based on a tagged training set. Medical doctors from Radiology Department at Hacettepe University Hospital tagged a subset of the radiology reports for important findings.

A tagging tool is provided to the doctors. Tagging consisted of selecting continuous phrases within report sentences. Each selection contained a complete finding; with its location and name. As the tagging tool provided only continuous phrase selection, a single selection might contain multiple findings with a single location in some cases, such as when conjunction words are involved. The tagging tool is actually developed for and used in a larger project, to match SNOMED concept identifiers with Turkish text.

The tagged phrases are analyzed for morphemes and string suffixes. Also, 2 words preceding and following tagged phrases are included in the analysis for detecting boundaries. Based on the frequencies of morphemes and common string suffixes, local templates are crafted.

In creating of local templates, it is assumed that a keyword (a morpheme, a string suffix or a special keyword) forms a meaningful relation between its left-hand-side and right-hand-side. There are exceptions where the local template might only create a meaning for its l.h.s. or r.h.s. component. There are also exceptions that have multiple keywords, and left-hand-side, central, and right-hand-side components.

The definitions of left-hand-side or right-hand-side do not pose any limitations on how far the text goes except for the beginning point of the text (which is always adjacent to the keyword). Central component has to be bounded by two keywords of the same template. How the boundaries of left-hand-side and right-hand-side components are determined will be explained in the following sections.

The local templates are grouped into different *individual meanings*, where each *individual meaning* can be instantiated by any of the local templates within its group. An *individual meaning* is defined as a single unit of information from a sentence. *Individual meanings* are shown as conceptual graph relations.

The first phase of the method, namely application of the local templates, is able to extract localized information from text. In order to extract non-local information such as a distributivity operation (by means of conjunction), rules based on Turkish grammar are used. These rules are applied on the conceptual graph of a sentence, and deduce new relations which are added to the conceptual graph itself. The rules are applied until no more rule matches the conceptual graph of the sentence. Only new relations (*individual meanings*) are appended to the graph, no morph or removal operation is performed. Thus, the algorithm may extract more information than present in a sentence, containing leftover information from previous steps.

The algorithm used is multi-phased, and the first phase requires correct morpheme identification. Due to the simplicity of the method only the morphemes of the last inflectional group are necessary. A simple morphological analyzer (Zemberek) and some workarounds are deemed sufficient.

Figure 4.1 shows the effect of morphemes on the sentence structure. In first example, *ACC* stands for accusative and *POSS3S* stands for 3rd singular person possession morphemes. In the second example, note that the locative suffix, *-de*, might extend to *hemoperitoneum* if it was an anatomical location.

4.2.1 Morphological Analysis and Workarounds

For the rule based algorithm, morphological analysis is performed using Zemberek library [1]. Zemberek analyzes a given word string by trying to regenerate it from a known list of roots,

Karaciğerin	konturları	, büyüklüğü	ve	parankim
liver+POSS3S	contours+ACC	, size+ACC	and	parenchymal

yapısı	normal olarak izlenmi stir.
structure+ACC	is normal.

Contours, size and parenchymal structure of liver is normal.

Hemaperitonium	,	mesane	lümeninde	hematom	ve	hava.
Hemoperitoneum	,	urinary-bladder	cavity+ACC+LOC	hematoma	and	air.

Hematoma and air in urinary bladder's cavity, hemoperitoneum.

Figure 4.1: Sample report sentences

morphemes, and rules governing agglutination. Zemberek, by design, may not try to generate all of the derived forms, but always generates the complete list of alternatives for inflections. Common derived forms of words were chosen to be included in its lexicon. This design choice makes its results less vulnerable to overly generating uncommon derivations. Zemberek is used as a syntax checker for Turkish and Turkic languages in many software packages.

In our case, regenerating the input word strings from a fixed dictionary of root words makes morphological analyzer vulnerable to medical terms that do not exist in its lexicon, but which are still used with agglutination in our reports. In order to overcome this limitation, a workaround is devised. Morphemes are made optional in the templates that match parts of sentences (first phase of algorithm). Thus, for most rules, two alternative forms exist; one that matches a morpheme analyzed by Zemberek, and a second form that uses the common surface representation of the morpheme.

Although we may use surface forms of morphemes, still using a morphological analyzer provides us with a priority selection. If a word can be analyzed by the morphological analyzer, then only the rules with morphemes are put into effect. Otherwise, rules with surface forms allows us fallback to string suffix matching. Turkish has complex morphotactics, and surface forms of even a single morpheme are affected by a variety of processes such as vowel harmony, vowel and consonant elisions or modifications in both morphemes and roots [29, 26]. This prioritization of trusting a morphological analyzer over checking string suffixes allows us to be more error-tolerant in selecting surface form suffixes to be included in the rules.

4.2.2 Form of Local Templates

Each template consists of a list of consecutive words. Each word *w* of the template will be compared to a word *x* in the sentence in order. If all words of the template match a consecutive part of a sentence, then that template is counted as a *match* at that point in the sentence.

Each template word *w* is one of the following:

- An string of three periods to match any word *x*
- A quoted string prefixed by a dash, which would match to string suffix of the previous word *x*
- A non-quoted string prefixed by a dash, which would match to any morpheme of the previous word *x* (the string denotes name of a morpheme)
- A quoted string suffixed by a dash, which would match to string prefix the word *x*
- A non-quoted string suffixed by a dash, which would match to morphological root of the word *x*
- The special string `[numb]`, which would match the word *x* only if *x* is numeric
- The special string `\$`, which would match to end of sentence (for detecting verbs)
- Any other string, which would match to *x* if the two strings are the same

Each template word may also be marked with an identifier, which would denote the relation type of the matching word *x* with the template. These identifiers are appended inside inequality signs < and > to the template string. These identifiers are essentially the names of the edges of relation nodes in the resulting conceptual graph. The same identifier may appear multiple times for different *x*, in which case concatenation would take place.

Table 4.3 contain example template strings showing different possibilities with the above definitions. Table 4.4 shows how example templates would match to some sentence parts.

Table 4.3: Local template examples

Template String (*w* separated by whitespace)
`...<bag> ile uyumlu<anlam> ...<bag2>`
`...<bag> ile uyumlu<anlam> görünüm<anlam>`
`...<nerede> -"de" ...<ne>`
`...<nerede> -"da" ...<ne>`
`...<nerede> -ISIM_KALMA_DE ...<ne>`
`...<baglanan1> ve ...<baglanan2>`
`...<nerede> -"den" ayrılan<iliski> ...<ne>`
`...<nerede> düzeyinde<iliski> ...<ne>`
`... azalmıştır<yuklem>` $

Table 4.4: Sample templates for some Individual Meanings

Type	Template	Matches to
LOC	..*(a)*.. +LOC ..*(b)*..	lümen+POSS3SG+LOC hematom
	..*(a)*.. +"de" ..*(b)*..	lümeninde hematom
CONJ	..*(c)*.. , ..*(d)*..	hematom , hava
	..*(c)*.. ve ..*(d)*..	hematom ve hava

4.2.3 Matching of Local Templates

The matching algorithm works on a sentence basis. The sentence is divided into tokens, each token *x* being a word, numeric value, or punctuation. Starting from every token *x*, every template is tried to be matched.

After matching of all possible templates is complete, boundary determination takes place. If a template begins or ends with an "any match" token (three consecutive dots), the other boundary of this element (far boundary from the template) needs to be determined. In order for boundary determination to take place, keywords of all matched templates are marked on the sentence tokens *x*. A keyword is any token of a template that is not an "any match" token. The marks can be positioned either on left half or right half or all of the token *x*. The mark is positioned on left half if only a prefix is matched. The mark is positioned on right half if the matched token is a suffix / morpheme. The mark is positioned on both halves if matched token is a whole word.

After keywords' positions are marked in the sentence, the gaps between keywords are concatenated together. These gaps map to "any match" tokens in templates. The templates efficiently show the relation between consecutive gaps. Each gap is a concept node in the conceptual graph of the sentence. The connecting templates become the relation nodes. The edges are named with the templates' identifiers.

The boundary determination process assures that medical terms consisting of multiple words can be grouped together as a single phrase when there is no template to relate them. Figure 4.2 contains an example on how the templates are matched and expanded to determine gap boundaries. The result of template-matching on the sample sentence can be seen in Table 4.5, and as a conceptual graph in Figure 4.3

```
        Hemaperitonium , mesane   lümeninde hematom ve hava
CONJ  ......................... , .........
LOC                                  .........de ............
CONJ                                             ............ ve ......
        ..........(I)............ ,  ...........(II)..........de ...(III)... ve ..(IV)..
```

Figure 4.2: Application of first phase templates

Table 4.5: Resulting relations after first phase

Relation	Concept 1	Concept 2
Conjunction	*(left)* Hemaperitonium	*(right)* mesane lümen
Location	*(where)* mesane lümen	*(what)* hematom
Conjunction	*(left)* hematom	*(right)* hava

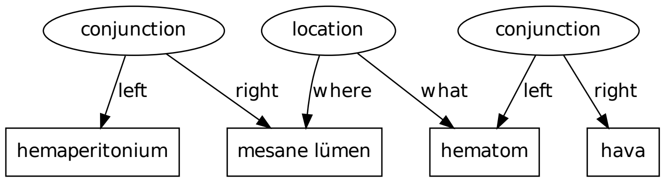


Figure 4.3: Resulting conceptual graph for the example sentence

4.2.4 Second phase: Non-local rules

The second phase of the algorithm works on the conceptual graph of a sentence. Second phase rules govern the expansion of the set with relations which exist between nonconsecutive phrases in the sentence.

Rules of the second phase specify two input relations, and a resultant relation. In each iteration, any two relations are selected from the input conceptual graph, *relation-A* and *relation-B*. If both *A* and *B* satisfy the conceptual rule, then a third *relation*, *relation-C* is instantiated and added to the graph. This process continues until no more expansions are possible.

Currently three conceptual rules are implemented. First two rules handle the left-distributivity and right-distributivity of *LOCATION relations* over *CONJUNCTION relations*. Conjunctions can be on the left-hand-side or right-hand-side of another relation, in which case the relation should be extended. The details of these rules can be seen in Figure 4.4 and Figure 4.5.

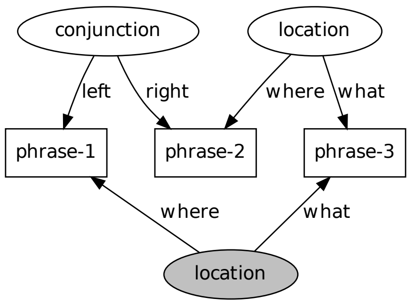


Figure 4.4: Conceptual graph rule for left-distributivity of Location over Conjunction

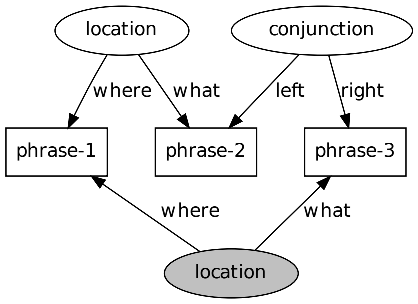


Figure 4.5: Conceptual graph rule for right-distributivity of Location over Conjunction

The third rule handles proper expansion of adjective phrases. These phrases take the form of a qualifier word followed by a phrase, and can be chained. When faced with these phrases, first phase templates will generate relations that are formed by the qualifier word and the right hand side phrase. Any qualifier that is in left hand side of this relation will be bound to the qualifier word instead of the correct r.h.s. phrase. These phrases are endocentric constructions [23], in which the rightmost phrase can be replaced for the entire phrase. Third rule will apply if *relation-B* has a *QUALIFIER* parameter, and *relation-A* has any parameter that is the same with *B*'s *QUALIFIER* – and it will instantiate a new *relation-C* which is a copy of *relation-A*, but the *QUALIFIER* replaced by the r.h.s. parameter of *B*. The details of this rule can be seen in Figure 4.6.

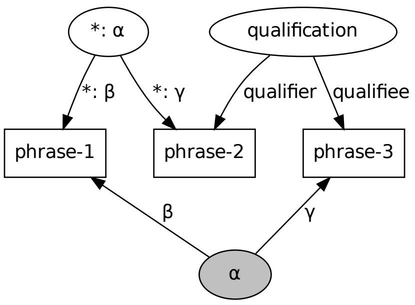


Figure 4.6: Conceptual graph rule for qualification relation chaining

Resulting set of relations after second phase can be seen in Table 4.6, and as a conceptual graph in Figure 4.7. Last two relations (marked with *) are deduced incorrectly due to missing medical information (*"Hemaperitonium"* is not an anatomical location).

Table 4.6: Set of relations after second phase

Relation	Concept 1	Concept 2
Conjunction	*(left)* Hemaperitonium	*(right)* mesane lümen
Location	*(where)* mesane lümen	*(what)* hematom
Conjunction	*(left)* hematom	*(right)* hava
Location	*(where)* mesane lümen	*(what)* hava
Location*	*(where)* Hemaperitonium	*(what)* hematom
Location*	*(where)* Hemaperitonium	*(what)* hava

4.3 Data-driven Method

A data driven method is devised in order to overcome the problems associated with hand-coded rules. This second method is constructed around dependency parsing and a rule learning algorithm to map from dependency graphs to conceptual graphs.

In order to create a conceptual graph from a sentence, first the sentence tokens are morpho-

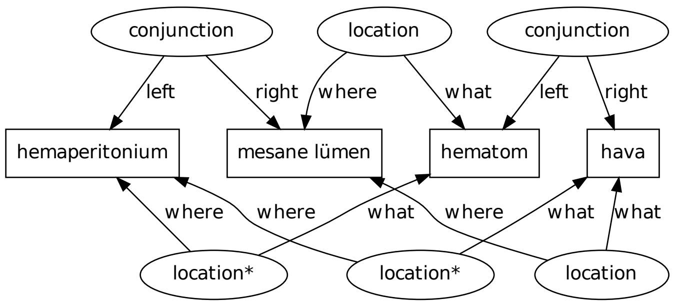


Figure 4.7: Conceptual graph showing relations after second phase

logically analyzed. The second step is to disambiguate the analyses. The third step requires a dependency parser to find out interword dependency relations within the sentence. The last step is construction of conceptual graph relations from dependency links using learned rules.

A training set is created for morphological disambiguation, dependency parsing and conceptual graph construction. As the method is data driven, this training set had to be smaller in breadth so that the different machine learning algorithms used in various steps may perform to their limits. The details and statistics of this training set can be found in Chapter 5.

The data-driven method learns rules based on tagged instances and dependency trees of sentences. The learned rules are different than first methods'. A rule structure consists of an acyclic undirected graph, with added constraints on its nodes and edges. This graph is matched to a target sentence's dependency tree. Whenever the constraints allow a match, named nodes of the graph determine the conceptual nodes within the sentence. Constraints on edges allow limiting the direction of the edge and edge's dependency label. Constraints on nodes specify a match-candidate word's morphological tags, part of speech tag of last inflectional group, and root lexeme.

Two algorithms are explored for automatic rule learning. The first one is a covering algorithm based on simulated annealing. The second one uses Support Vector Machines to learn the binary constraints on rule structures.

4.3.1 Morphological Analysis

Due to its limitations outlined in Chapter 3, Zemberek library is found to be a poor choice for generating sufficiently detailed analyses for dependency parsing. TRMorph [4] is chosen for the task because it is based on a finite state transducer framework. TRMorph uses a heavily modified version of Zemberek's lexicon.

TRMorph is capable of extracting derivational morphemes and the resulting POS tags after each derivation. Still, the morphological analyses do not have one-to-one correspondence with outputs of other morphological analyzers used in previous works in dependency parsing of Turkish. Although this does not pose an important problem for inflectional morphemes, differences in derivational boundaries might result in incomparable results. In order to stay on common ground with previous works, a postprocessing step is added to the results of the analyzer. Some morphemes, which TRMorph does not mark as derivational (and thus does not generate new POS tags), are marked as derivational boundaries and replaced by new POS tags. This group of postprocessed morphemes consist of non-finite verb markers; such as converb markers (form subordinating clauses with adverbial function), participle markers (make non-finite verbs of relative clauses) and verbal noun markers (form noun clauses from non-finite verbs).

4.3.2 Morphological Disambiguation

Due to the relatively small training set size, purely statistical methods for disambiguation would be inappropriate for this study. For this reason, Yuret and Türe's rule learner based disambiguator is preferred.

Although Yuret and Türe publicly provide a trained model for disambiguation, this model was inappropriate for our requirements. The morphological analyzer outputs used by the published model was different than TRMorph's. The second reason was the content of their training set, which does not consist of any radiology reports. Considering that the decision lists are structured around various string suffixes of nearby words, the available trained model would perform poorly on our content which is very different.

4.3.3 Dependency Parsing

MaltParser is used for training and parsing utilizing a support vector machine based model. In order to configure MaltParser in this study, we used the parameters used by Eryiğit et al. for CoNLL-X shared task [2]. These parameters are publicly available. This availability provided us with the option of training the parser with our own training set. Although the parameters were optimized for Turkish Treebank, due to our training set's very small size, we did not attempt any further optimization. Motivated by Eryiğit et al., we used IG-to-IG links for dependency parsing.

4.3.4 Learning Rules with Simulated Annealing

In order to find conceptual relations using dependency links, a rule-learning covering algorithm is implemented. This algorithm is similar to [30, 31]. The covering algorithm tries to find minimum number of rules that cover the entire training set. Each learned rule is in the form of an acyclic graph with constraints on its nodes and edges. If a learned rule aligns with the dependency tree of an input sentence and if all constraints match, some of the aligning words and rule's subject are used to construct a conceptual relation.

The input to the learned rules is the labeled inter-word dependency graph of a sentence, and morphological features of every word. The output of a matching rule is a single conceptual relation, with labeled edges between two or more phrases. The output phrases can span multiple words, and are delimited (starting and ending words are known, unlike the templates of the rule-based algorithm). A sample input dependency graph is shown in Table 4.7. Its expected output relations are shown in Table 4.8 as separate entitites. The conceptual graph of the expected relations as a whole can be seen in Figure 4.8. Note that some of the information in the sentence cannot be represented by the output conceptual graph, and is lost.

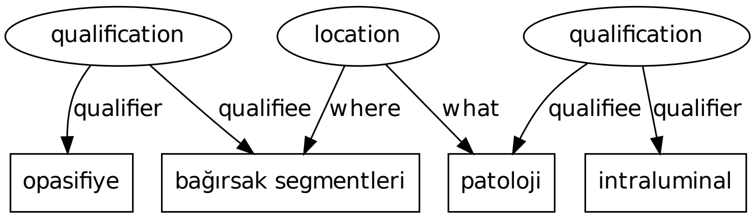


Figure 4.8: Conceptual graph representation of the expected output relations

The algorithm selects a random positive instance from the corpus and tries to generate a matching rule for it. Rules which are more generalized and which cover more positive training instances and less negative instances are preferred. For the rule learning algorithm, tagged

Table 4.7: A sample input sentence, as a dependency graph and morphological features

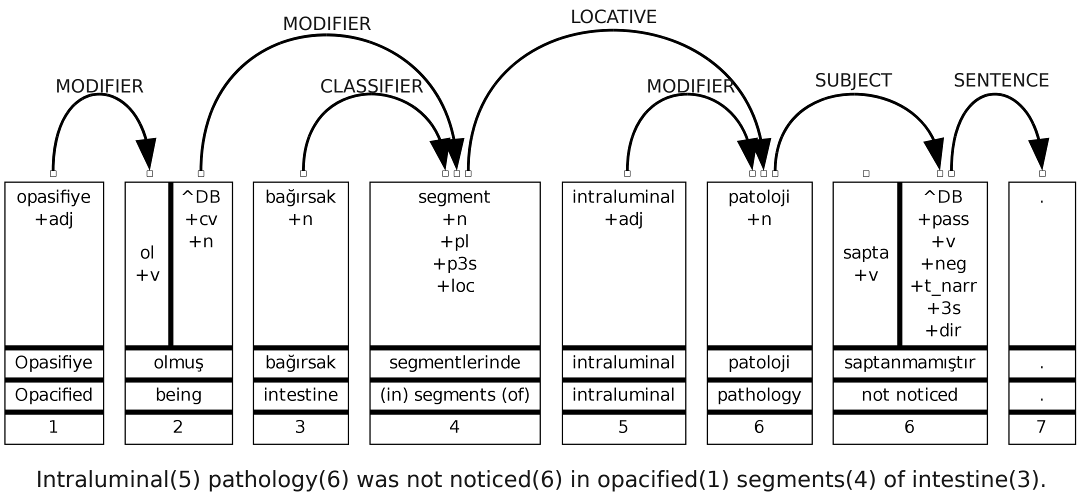


Table 4.8: Expected output conceptual relations after processing the sample input sentence

Relation	Concept 1	Concept 2
qualification	qualifier=1	qualifiee=3,4
qualification	qualifier=5	qualifiee=6
location	where=3,4	what=6

relations form the positive instances. Rest of the corpus forms the negative instances. The algorithm continues until all positive training instances are covered, thus forming a covering algorithm.

The covering algorithm first marks the entire training set as not-covered. In each main iteration, it first selects a not-covered training instance as seed, and generates the *most specific rule* for matching that instance using the dependency tree of the sentence. Iterations start with an initial rule based on this *most specific rule*. In each iteration, current rule is constantly updated, either by creating slightly relaxed or slightly stricter versions of it.

A rule consists of a subject (name of the conceptual relation), and *n* nodes. Each node represents a word, and is connected to a parent node in the rule's graph. The rule is actually a subtree of the dependency graph, but directed edges and word nodes are replaced by constraints. Each node have constraints on the lexeme (root of the word), existing morphological features, non-existing morphological features. Lexeme constraint and individual morphological feature constraints may be omitted. Each edge (between a node and its parent) can have two constraints; a label constraint (matching the name of the dependency link between two words), and a direction constraint (directed from node to parent, or no direction is imposed). Both the label constraint and direction constraint can be omitted.

The *most specific rule* matching a conceptual relation is the exact subtree of the dependency tree, which contains only all the words in the relation, and required nodes to connect them as a tree if any. This rule will have label and forward direction constraints on all its edges, and will contain the full copy of the morphological features (both existing and non-existing) of each word, and the words' roots. A sample *most specific rule* can be seen in Figure 4.9. Note that the universe of morphemes shown in this figure contains only 6 morphemes.

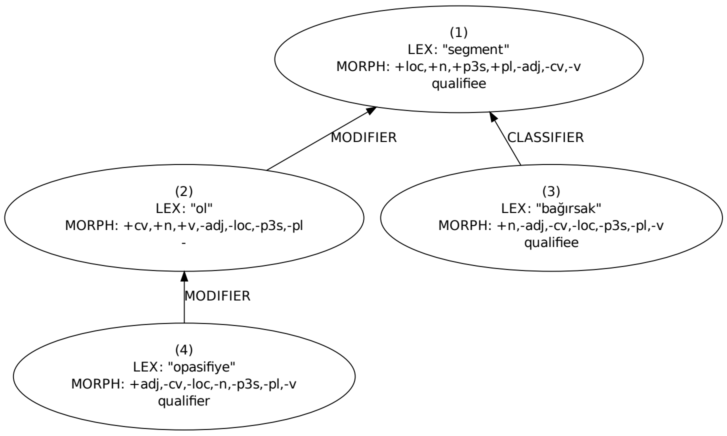
The *most general rule* matching a conceptual relation is the completely relaxed version of the *most specific rule*. Dependency label and direction constraints are removed (so either child or parent node may be the modifier or head in the dependency graph), all required-to-exist and required-to-be-absent morphological constraints are removed, and the lexeme's root constraint is removed from nodes. *Most generalized version* of the rule in Figure 4.9 can be seen in Figure 4.10.

Each node in the rule may have a conceptual relation edge name attached. If a conceptual relation edge name is attached, the aligned word with that node will get connected to the conceptual relation on that specified edge. If an edge name is not attached, then an aligned word will not be used in the conceptual relation (but is required to exist in the sentence as a connecting component).

In the learning process, the rule only gets transformed between its most general and most specific versions. An in-between version of the most specific and most generalized rules can be seen in Figure 4.11.

ID	Parent	Dir	Label	Lexeme	Morph Tags	Absent Tags	Concept
1	.	.	.	segment	loc,n,p3s,pl	adj,cv,v	qualifiee
2	1	fw	MODIFIER	ol	cv,n,v	adj,loc,p3s,pl	.
3	1	fw	CLASSIFIER	bağırsak	n	adj,cv,loc,p3s,pl,v	qualifiee
4	2	fw	MODIFIER	opasifiye	adj	cv,loc,n,p3s,pl,v	qualifier

(a)


(1)
LEX: "segment"
MORPH: +loc,+n,+p3s,+pl,-adj,-cv,-v
qualifiee
MODIFIER
CLASSIFIER
(2)
LEX: "ol"
MORPH: +cv,+n,+v,-adj,-loc,-p3s,-pl
-
(3)
LEX: "bağırsak"
MORPH: +n,-adj,-cv,-loc,-p3s,-pl,-v
qualifiee
MODIFIER
(4)
LEX: "opasifiye"
MORPH: +adj,-cv,-loc,-n,-p3s,-pl,-v
qualifier

(b)

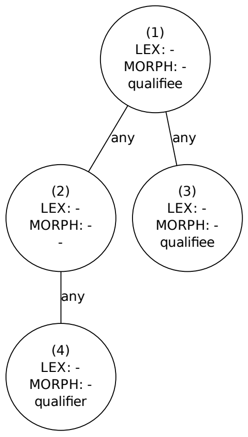
Figure 4.9: Most specific version of a rule

Every node in a rule contains 46 morphemes, 1 lexical form, 1 dependency link label, 1 dependency link direction. Each of these constraints may be omitted or kept. A typical rule structure has 2 to 4 nodes, which makes the branching factor on the order of hundreds. Simulated annealing is used for learning the rule with most general constraints. This is preferred over alternatives such as beam search due to the large branching factor of the rules.

The pseudocode for rule learning process can be seen in Figure 4.12. Two levels of iterations are performed. In the first level, only the dependency link directions and dependency link

ID	Parent	Dir	Label	Lexeme	Morph Tags	Absent Tags	Concept
1	.	.	-	-	-	-	qualifiee
2	1	-	-	-	-	-	.
3	1	-	-	-	-	-	qualifiee
4	2	-	-	-	-	-	qualifier

(a)



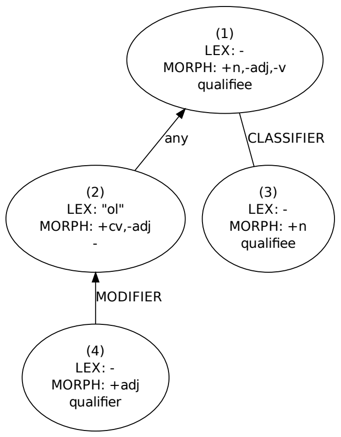
(1) LEX: - MORPH: - qualifiee
any
any
(2) LEX: - MORPH: - -
(3) LEX: - MORPH: - qualifiee
any
(4) LEX: - MORPH: - qualifier

(b)

Figure 4.10: Most generalized version of a rule

ID	Parent	Dir	Label	Lexeme	Morph Tags	Absent Tags	Concept
1	.	.	.	-	n	adj,v	qualifiee
2	1	fw	-	ol	cv	adj	.
3	1	-	CLASSIFIER	-	n	-	qualifiee
4	2	fw	MODIFIER	-	adj	-	qualifier

(a)



(1) LEX: - MORPH: +n,-adj,-v qualifiee
any
CLASSIFIER
(2) LEX: "ol" MORPH: +cv,-adj -
(3) LEX: - MORPH: +n qualifiee
MODIFIER
(4) LEX: - MORPH: +adj qualifier

(b)

Figure 4.11: A rule that is only partially generalized

label constraints are tried to be optimized. This level starts with the most specialized form of the rule with respect to these two features. All morphological features and lexeme is kept in most generalized form during this process. In the second level, only the morphological constraints and lexeme constraints are optimized. This level starts with most generalized form of these constraints.

Selection of a two level optimization process is the result of an analysis on algorithm's convergence performance. Number of dependency link direction and dependency link label constraints are very low compared to number of morphological constraints. But dependency constraints play a role nearly as important as all of the morphological constraints. Trying to optimize all of these constraints in the same iteration resulted in decreased convergence.

The iteration part is a slightly modified version of the simulated annealing algorithm. In each iteration, k slightly specialized or generalized versions of the current rule is generated with equal probability. This is similar to a beam search. All of these versions are tested on the training set. Of all the generated candidate candidates, only a single one with the least error is selected as the candidate. If multiple candidate candidates share the same error, a random one is selected as the candidate.

The chosen candidate differs with the current rule on only a single constraint's existence. The candidate rule is tested on the training set. Based on an error metric and a generalization metric, the candidate rule is marked as better than or worse than the current rule. If the candidate rule is better, current rule is updated with a constant high probability ρ. If candidate rule is worse than the current rule, current rule may still be updated with a lower probability $1 - \rho$. Generalization metric is the number of removed constraints. The *most specific version* of a rule contains many redundant morpheme constraints, which causes exactly equal error rates. The generalization metric acts as a tie-breaker in these cases and promotes more general rules over specific ones.

When the iteration limit α is reached, the search stops and the best rule found so far is returned. Also another iteration limit β resets the current rule to the best rule if the search cannot find a better rule in a limited neighborhood of it. Although this limits the maximum radius of the local-minima that may be circumvented by simulated annealing, our experiments showed that the learning algorithm tends to get stuck after drifting too much away from the current best rule, failing to find any better rule until α is reached.

Impurities and noise within the training set need to be handled properly. Otherwise overfitting to the training set might occur. A threshold exists on minimum number of covered examples by a single rule. No rule covering less than this many examples will be added to the global set of learned rules.

In an ideal scenario, simulated annealing method does not need generation of k new candidates. But the error function consists of very large flat regions in our case. Most of the time, randomly changing a single feature does not affect the value of the error function. This might be circumvented by changing multiple features at once in each iteration, starting with more changes at initial iterations, and ending with single changes. Instead, we chose to change a single feature at a time, but employ a method similar to beamsearch. This resulted in less training parameters to be tuned for the application.

The iterations limits varied linearly with the number of nodes in the *rule structure*. For the first level of iterations, an iteration limit of $10*(numberofnodes)$ is used. Best iteration limit is set as constant 10. Branching constant is set as 2. For the second level of iterations, an iteration limit of $50*(numberofnodes)$ is used. Best iteration limit is set as constant 100. Branching constant is set as 5. Number of features affected choice of the parameters. Repeated tests are conducted on the same training set to find the parameters that lead to less variance between results of successive runs, resulting in convergence.

4.3.5 Learning Rule Constraints with Support Vector Machines

When examined with a broader perspective, the intermediate constraint selection steps between *most general rule* and *most specific rule* can be seen as a classification problem with binary features. Support vector machines provide a proven method for solving such a problem.

In order to apply binary classification solutions to the problem, first the mapping between rules and vector space representation must be made. In our solution, one *most generalized rule* for each positive training instance is created and duplicate copies are removed. These *most generalized rules* will be called as *rule structures* in this subsection. Separate binary support vector classifiers are trained for each *rule structure*. Thus, only the simulated annealing algorithm of the previous subsection is replaced with support vector machines, but the

```
FUNCTION LearnRules(TrainingSet)
    Mark all examples as not-covered
    Rules <- empty
    DO
        Seed <- Randomly select from not-covered examples
        Rule <- LearnRuleFromSeed(TrainingSet, Seed)
        Test(Rule), mark all examples the rule covers as covered
        Append Rule to Rules
    UNTIL (All examples are covered)

    RETURN Rules
ENDFUNCTION

FUNCTION LearnRuleFromSeed(TrainingSet, Seed)
    MostSpecificRule <- Generate most specific rule from Seed

    Rule <- Remove all morphological constraints of MostSpecificRule

    Rule <- SimulatedAnnealingSearch_OnDependencyConstraints
                (Rule, TrainingSet, IterationLimit_D, IterationLimitBest_D, BranchingConstant_D)

    Rule <- SimulatedAnnealingSearch_OnMorphologicalConstraints
                (Rule, TrainingSet, IterationLimit_M, IterationLimitBest_M, BranchingConstant_M)

    RETURN Rule
ENDFUNCTION

FUNCTION GeneralizeRule(Rule, N)
    Create N copies of Rule
    Remove 1 random feature constraint on each copy
    RETURN Copies
ENDFUNCTION

FUNCTION SpecializeRule(Rule, N)
    Create N copies of Rule
    Add 1 random feature constraint on each copy
    RETURN Copies
ENDFUNCTION

FUNCTION SimulatedAnnealingSearch_X(Rule, TrainingSet, IterationLimit, IterationLimitBest, BranchingConstant)
    BestRule <- Rule
    NumIterations <- 0
    NumIterationsBest <- 0

    WHILE (NumIterations < IterationLimit)
        IF (Random() >= 0.5)
            Candidates <- GeneralizeRule(CurrentRule, BranchingConstant)
        ELSE
            Candidates <- SpecializeRule(CurrentRule, BranchingConstant)

        Test Candidates on TrainingSet
        Candidates <- Select candidates with lowest error

        Candidate <- Randomly select from Candidates
        Test(Candidate) on TrainingSet

        IF Candidate has less error than Rule (or is more generalized if tie)
            TestProb <- p
        ELSE
            TestProb <- 1-p

        IF Random() >= TestProb
            Rule <- Candidate

        IF Rule has less error than BestRule (or is more generalized if tie)
            BestRule <- Rule
            Reset NumIterationsBest

        IF NumIterationsBest > IterationLimitBest
            Rule <- BestRule
            Reset NumIterationsBest

    ENDWHILE

    RETURN BestRule
ENDFUNCTION
```

Figure 4.12: Pseudocode for rule learning

rule representation is almost the same.

Rule structures by themselves do not convey any criteria on inclusion or exclusion of features. Rather, they separate the data into orthogonal categories. Each *rule structure* is aligned with all subtrees of sentences in the dataset, and for every aligning subtree a positive or negative vector instance is created. Positive or negative is determined by *rule structure*'s internal information on words aligning with which nodes will become the relation's concepts.

The vector instances created by a *rule structure* contain features of every word aligning with a node of the structure. Thus, by definition, the vector instances created by different *rule structures* do not have to be the same length, and even if they are, are not comparable. This is the reason of training a separate support vector machine per *rule structure*. Note that this does not require every tagged relation instance be matching to different *rule structures*. Different *rule structures* may align correctly with the same tagged relation (correctly with respect to conceptual relation edge labels).

For every word aligning with a node, a group of binary features are appended to the vector. These features are as follows:

- Is the dependency tree edge forwards (with respect to *rule structure*'s layout, between aligned node and aligned node's parent node)
- Is the dependency tree edge backwards (with respect to *rule structure*'s layout, between aligned node and aligned node's parent node)
- One feature per each dependency relation label (categorical feature, only the aligned label's feature has a value of 1, and others' features have a value of 0)
- One feature per each possible morpheme (if morpheme exists in aligned word then 1, if morpheme does not exist then 0)

The need for specifying non-existant morphemes in the previous subsection does not apply to support vector machine model. The previous subsection focuses on finding a set of required criteria, whereas support vector machines are fed with the values of features and are supposed to find the required criteria themselves.

The main difference between *rules* of the previous subsection and *rule structures* lies in where the information on individual instances are kept. A *rule* holds exact values of the features in its structure (value are imposed by its *most specialized* variant), and the alternative instances only state which features needs to be kept or removed. On the other hand, a *rule structure* only holds the possible features but not their values; the values of features are stated differently in each instance that is aligned with the *rule structure*.

The root lexemes of aligned words could also be represented as categorical features, but are chosen to be omitted in order to prevent the vector representation becoming too large.

As mentioned previously, a single relation (whether tagged or not-tagged) does not need to belong to a unique *rule structure*. Even, a single relation does not have to have a single representation in the vector space. A single *rule structure* may align with the same words in different ways. If the conceptual relation edge labels aligning with the words stay the same, the same conceptual relation might be represented by a variety of vectors.

Each *rule structure*'s representation vector size increases linearly with respect to its number of nodes. Number of instances in each dataset per *rule structure* increases exponentially with respect to its number of nodes. Combined with the requirement to train a different support vector machine per *rule structure*, this led us to using linear support vector machines.

4.3.6 Variations on the Algorithm

4.3.6.1 Inclusion of root lexeme

The reports processed contain a vast amount of medical terms. Most of these terms are not addressed in morphological analyzer, and mostly get assigned with *noun* or *adjective* POS tags. Assigning a medical term with such a coarse POS tag may sacrifice performance of extraction. A more suitable method would be to be able to assign more semantic POS tags to these words at morphological analyzing step, such as *body structure*, *adjective modifying findings*, *adjective specifying location with respect to body structures*, although this would contradict with the main objective of this research.

Inclusion of the root lexemes of words within rules may allow the algorithms to learn better rules in absence of medically semantic tags attached to words. The algorithms might be able to relate root lexemes with the rules, rendering the need for medical information on words unneccessary.

On the other hand, inclusion of the root lexemes might cause the algorithms to overfit to training data, failing to generalize to unseen instances in test set.

4.3.6.2 Inclusion of string suffixes of root lexeme

An alternative to including root lexeme is including string suffixes of the root lexeme. Most medical terms, especially adjectival terms, are derived from Latin in medical reports, and are easy to identify as modifiers only checking the last two or three letters of the word.

The disadvantage of this approach is the effect on the vector size. String suffixes need to be represented like the morphological tags, one separate value for presence and absence of every possible string suffix in the dataset.

4.3.6.3 Appending one additional parent node to rule representation

Although a rule representation graph has limiting boundary conditions on its leaves regarding their dependency relations, this is not the case for the root node of the representation. Especially in the case a related concept word becomes the root node of the representation, the rule representation does not place any constraints on the role of the word in the dependency graph regarding its dependency heads.

Instead of generating *most specialized rules* with the smallest subtree that connects all related concepts, one more node, the dependency head of the root node can be appended to the rule graph as a new node. This would be only needed in the case where the root node plays role in the conceptual relation.

Appending this additional node would make all rule representations being rooted with a word that does not play a role in the conceptual relation. Dependency constraints may be placed on *all* words playing role in the conceptual relation.

The advantage of this variation would be preventing larger relations being matched with smaller relations. The additional root constraint would prevent the match in these cases.

4.3.6.4 Disallowing zero-errors

An important consideration when implementing learning algorithms is to prevent overfitting to training set. This can be accomplished in many ways, such as separating some of the training set, and using that part only for tuning. Another method would be to use k-fold cross validation on training set itself, in order to find the best parameters of the algorithm that would not cause overfitting.

In the simulated annealing algorithm, overfitting occurs when the best value of the error function is reached during training, one match and zero errors. After this value is reached, the algorithm will not be able to replace the cached *best* rule with a rule that does not overfit. We can resolve this problem by disallowing candidate rules with zero-errors.

In our algorithm, overfitting problem does not occur so seriously unless root lexemes are included in rule representations.

In the support vector machine learning algorithm, overfitting is prevented by k-fold tuning for parameters.

4.3.6.5 Filtering of learned rules

After training, learned rules covering less than a given threshold of positive training instances may be filtered out. These rules would most likely not match any positive instances in test set, but may drop performance by matching to random negative examples.

Another filtering technique is filtering based on error rate. If error rate of a learned rule is higher than a threshold (measured in the training set), the rule may be discarded before testing. But this technique may not cause any performance gains as the learning algorithm's target is to minimize error rate.

4.3.6.6 Filtering of data

Both learning algorithms may suffer from the sparsity of positive training instances, and this may be amplified by the orthogonal separation of training instances by the choice of rule representation. By definition, a *most generalized rule* or *rule structure* to match *n* words will not match *m* words if $m \neq n$. This orthogonality acts to make the training data even sparser.

This problem affects the support vector machine learning more than the simulated annealing method. A support vector machine is trained as a binary classifier, and in order to be trained properly, it needs to have enough positive and negative training examples.

In order to filter data, *most generalized rules* are generated for every positive data instance and duplicate copies are removed. These form the *rule structures*. Each individual positive data instance is marked for which *rule structures* it may match with (a data instance may match with multiple *rule structures*).

After number of instances falling within each *rule structure* group is found, *rule structures* that contain less than a threshold of positive examples can be filtered out. After the *rule structures* are filtered, any positive data instance that belongs to no *rule structure* is removed from the dataset.

This procedure is applied on the whole dataset, not only training data. This procedure's purpose is not to train better on data, but rather provide better measurement accuracy. The numeric details on the procedure can be found in Chapter 5.

CHAPTER 5

EVALUATION

The evaluation of the proposed methods are done in two contexts: information retrieval and information extraction. For this purpose, two different datasets are used. Reports in both datasets are drawn from the master dataset mentioned previously in Chapter 4.

The first dataset consists of a set of radiology reports in different topics and a set of search queries. The search queries are free-text, written in a style similar to in reports. However, search queries are not full sentences, but partial sentences. Each search query contains only a few conceptual relations such as a finding, a location, and qualifiers for the finding and location. Each query has a defined set of reports it should return. This dataset was prepared by a medical doctor. This set is used for measuring performance of the rule based method in a retrieval context.

The second dataset consists of a set of abdominal radiology reports. The text of these reports are tagged for morphological disambiguation, dependency trees, and conceptual relations. This set is used for measuring performance of rule based and data driven methods in an extraction context. For the rule based method, this dataset is directly used as a test set. For the data driven method, this dataset is used for both training and testing purposes, utilizing k-fold cross validation to prevent training on test data.

5.1 Information Retrieval

Only the rule based algorithm is tested in information retrieval context. Also a baseline search algorithm is implemented and tested for comparison.

Rule based algorithm generates a set of *individual meanings* or *conceptual relations* from a given input sentence. Thus it extracts some of the information into structured form. In order to use these *individual meaning sets* in a retrieval scenario, a method similar to vector-space model is used.

First every document of the test set is parsed with the algorithm, and the extracted *individual meaning sets* are recorded into a database. The query strings are free text, written in a style similar to report sentences. In order to find relevant documents to a query, the query strings (which are similar to partial sentences) are assumed as a sentence on their own, and are parsed with the algorithm. The resulting *individual meaning set* is the query set. Intersection of this set with the sets stored in the database is used to retrieve relevant documents.

There are various alternatives for deciding if a document should be retrieved based on the intersection. The first alternative is to check if at least one *individual meaning* intersects between the whole document's set and the query set. A second alternative is to calculate the intersection with the whole document, but require the size of the intersection set to be greater than a threshold ratio of the search set. The third alternative is to calculate the intersection with only a single sentence in the document (thus retrieving relevant sentences instead of documents), and require a threshold.

5.1.1 Baseline Algorithm

In order to measure the algorithm's performance, a baseline comparison is required. A baseline algorithm is implemented for this purpose. The baseline algorithms makes a set intersection based search like the rule-based algorithm, but the sets consist of individual query and document words (root lexemes), not *conceptual relations*.

The same three alternative methods mentioned above apply to baseline algorithm as well. The algorithm can decide a match if a threshold of query words exists in the candidate document, or in a single sentence of the candidate document. Setting a threshold of zero would make the algorithm return a document if it contains any of the words in the query. Setting a threshold of one would make the algorithm return a document if it contains all of the words in the query.

5.1.2 Experimental Results

The test set consists of radiological reports on different topics. There are 53 reports and 100 queries in this set, forming 5300 data points. All the reports that a query must return are marked. If a report is not marked for a query, then that report should not be returned, and counted as an error if is returned.

Table 5.1 shows the result for all three alternatives of the rule based method's application, compared to varios configurations of baseline retrieval algorithm. The baseline algorithm has a very high recall in all cases, but a precision close to zero. The reason lies in the structure of the queries. A query string is a partial sentence written in a similar style to report sentences. Thus, it contains all the words that would be existant in a target sentence. The reason for very low precision is the query strings containing all generic words like *cyst*, *liver*, *lesion*. Most of these words exist in all sentences, even if they are not related together. The proposed algorithm, on the other hand, focuses on the relations of the individual words, and makes a compromise between very low precision and very high recall. As the query strings mostly contain one or two relations, threshold setting does not affect the results of rule based algorithm as much as anticipated. This is also the reason why more threshold values were not tested.

Table 5.1: Experimental results for rule-based method in retrieval context

Method	Match target	Threshold	Recall	Precision
Baseline	Whole report	Single match	96.05%	5.15%
Baseline	Whole report	50%	96.05%	5.42%
Baseline	Single sentence	50%	96.05%	5.46%
Rule Based	Whole report	Single match	67.10%	28.65%
Rule Based	Whole report	50%	65.78%	31.84%
Rule Based	Single sentence	50%	65.78%	31.84%

5.2 Information Extraction

A second set is used for information extraction context. In this set, outputs of both methods are analyzed and compared.

The second dataset consists of only abdominal radiology reports. Sentences in *clinical in-*

formation, *findings* and *conclusion* sections of the reports are manually tagged for existing conceptual relations. This dataset is also tagged for morphological disambiguation and dependency parsing for the data driven algorithm.

Initially the provided reports mostly contained sentences that show normality in the *findings* section. Most of these sentences were exactly the same. Although they contained some location relations (e.g. *Liver structure and surrounding area is normal*), their content was not interesting for information extraction purposes. Thus, after the tokenizing process, any sentence that has multiple exact copies in the dataset is completely removed including itself. Table 5.2 show the number of initial sentences and number of sentences after filtering for duplicates. As shown in the table, average number of sentences per report dropped significantly after filtering. However, the information in completely removed sentences is not lost, because similar versions of these sentences are not removed (e.g. *Liver structure and surrounding area is normal (except for the previously mentioned lesions)*).

Table 5.2: Effects of duplicate sentence removal

	Total number of sentences	Number of sentences per report
Initially	506	15.8
After removing duplicates	251	7.8

Only a subset of the relation types of rule-based method are tagged in this dataset. The reason is that the rule based method makes use of intermediate relations to deduce others, whereas in an extraction scenario, our target is more limited. Tagged relation types and their distribution can be seen in Table 5.3. A subset of the dataset can be found in Appendix B.

Table 5.3: Distributions of relations in 2^{nd} dataset

Relation type	Number of instances	Number of sentences
Location	159	110
Modifier	270	136
Locative modifier	91	67
Measurement	37	35
Other	9	5

5.2.1 Definition of Match and Error

In applying the proposed algorithms in information extraction context, first what counts as a match and what counts as an error must be defined. Given a sentence, both rule based and data driven algorithms extract a set of relations in the sentence. The relations are typed, and have labeled edges to different concept phrases in the sentence. Each related concept phrase may contain any number of words, with the only requirement of containing consecutive words within the sentence.

Match and error in information extraction may be defined differently depending on context. The problem is different than a classification or document retrieval problem, because extractions with partial phrases, extractions with excessive phrases (phrases containing more words than the truth), and conflicting extractions are possible.

Except from partial and conflicting extractions, choices in tagging of the dataset also affects definition of match and error. A *closed world assumption* will require any extracted data that does not coincide with any tagged instance to be marked as an error, similar to a classification problem. If only the interesting examples are tagged in the dataset, a *closed world assumption* may result in low performance measurements. In these cases, a better approach would be to check if an extracted data conflicts with tagged data, and mark as an error only when it conflicts. If the extracted information does not conflict or coincide with tagged instances, then it can be safely ignored.

In following experiments, match and error are defined in three different ways, and measurements are reported with all the calculated values.

A correct match (*true positive*) is defined as an extracted relation matching exactly to a tagged relation. The relation type, relation edge names, and the conceptual phrases must be exactly the same. If any of the extacted conceptual phrase is a subset or superset of the respective relation's conceptual phrase, the extraction is not counted as a correct match. Similarly, if any of the tagged relations is not exactly extracted by the algorithms, this will be counted as a *false negative*.

An incorrect extraction (*false positive*) is reported in three different forms:

Closed-world assumption If an extracted relation is not a match (*true positive*), count it as an error (*false positive*).

Closed-concepts assumption Consider an extracted relation as an error (*false positive*) only if at least one of its phrases intersects with a tagged relation. Otherwise ignore it.

Closed-relations assumption Consider an extracted relation as an error (*false positive*) only if at least two of its phrases intersect with tagged relations. Otherwise ignore it.

The first alternative, *closed-world assumption*, assesses the extraction problem as a classification problem.

The second alternative, *closed-concepts assumption*, is a relaxation on the first one. It is based on the assumption that, if any concept is tagged in the dataset with some relations, then those relations are all relations of that concept, and no additional relations (of that concept) exist. For other possible concepts, the dataset is not tagged for, and nothing can be said about their existence or absence.

The third alternative, *closed-relations assumption*, is another relaxation on the second one. It is based on the assumption that, if any concept is tagged in the dataset with some relations, then those relations do not have to be all relations of that concept within the sentence, and additional relations (that were of no interest for tagging purposes) might also exist.

Based on the above definitions for match and error, five criteria are reported on each experiment. The first one is *true positives*, which will be referred to as TP. Only a single definition of *match* is used, so this does not get affected by definition of error. The second one is *false negatives*, shown by FN. The third one is *false positives based on closed-word assumption*, which will be referred to as FP_{CWA}. The fourth one is *false positives based on closed-concepts assumption*, which will be referred to as FP_{CCA}. And the last one is *false positives based on closed-relations assumption*, which will be referred to as FP_{CRA}.

In addition to these values, one *recall*, and three alternative *precision* values calculated from the above values are reported. Additional measures such as *F1-score* will not be reported in order to not confuse the reader by more values.

5.2.2 Evaluation of Rule Based Method

Results of rule based method can be seen in Table 5.4. The results are lower than expected. This is mostly caused by incorrect extraction of phrase boundaries.

Table 5.4: Extraction results of rule-based method

	TP	FN	FP			Recall	Precision		
			CWA	*CPA*	*CRA*		*CWA*	*CPA*	*CRA*
Rule-based (Test-0)	137	289	752	539	285	32.2%	15.4%	20.3%	32.5%

5.2.3 Evaluation of Data Driven Method

All measurements are made using k-fold cross validation. For each experiment the dataset is separated into k folds of equal size. The separation is done on set of sentences, not tagged conceptual relations. Although the number of sentences are equal, number of tagged conceptual relations differ between folds. The same separation boundaries are used in all experiments.

For every reported experiment, 1 fold is removed as test set, and training is performed on the remaining $k - 1$ folds. Then the results of training is tested on the test set. This process is repeated k times for each fold, and the resulting TP, FN, FP values are summed. The precision and recall numbers reported are based on the accumulated TP, FN, FP results.

In most experiments $k = 10$ is used. In some experiments $k = n$ is used, where n is the number of data instances, resulting in leave-one-out cross validation. Exceptions that use a k with a different value than 10 are specified in the text and tables.

Unless specified in text, the given measurements for data driven method are results of tests on manually tagged morphological analyses and manually tagged dependency relations. The reason is to hide effects of accumulated errors caused by automated morphological disambiguation and dependency parsing. Measurements that include automated morphological disambiguation and dependency parsing are given at the end of the section.

5.2.3.1 Filtering of dataset

The rule representation chosen for the data driven method places data instances into orthogonal groups. Although some of the groups may share some data instances, *rule structures* that do not have the same relation type will not share any data instances. Moreover, even if two *rule structures* have the same relation type, they will not share any data instances if the concept labels are different (some relation types may have missing concepts labels, such as a measurement), or if number of words in the same labeled concepts differ.

In order to see the effect of rule representation, the following experiment is performed on the *whole* dataset. First, *most specialized rules* are generated for all the tagged instances. Then these *rules* are generalized and converted into *rule structures*, which are acyclic undirected graphs with labels on their nodes which align with concepts. Duplicate *rule structures* are removed. Number of tagged instances and number of unique rule structures are shown in Table 5.5. Number of unique *rule structures* per relation type is shown in the same table. Figure 5.1 shows the histogram of number of tagged instances vs. *rule structures*. X axis shows number of tagged instances and Y axis shows number of *rule structures*. Note that one tagged instance can be aligned with multiple *rule structures*.

Table 5.5: Tagged instances and Rule structures

Name	Instances
Tagged instances	566
location	159
modifier	270
locative modifier	91
measurement	37
Others	9
Unique *rule structures*	87
location	26
modifier	25
locative modifier	16
measurement	12
Others	8

As the dataset is divided orthogonally into different *rule structures*, number of available positive training examples per structure drops significantly. In order to minimize this divergent effect, the training data is filtered for tagged instances whose coverage is low within the *rule* *structures*. If any structure holds less than k positive examples, these structures are removed. After removing of *rule structures*, the training examples that don't belong to any rule structures are removed from the dataset.

$k = 2$ and $k = 10$ are chosen for the experiments. Unless otherwise stated, experiment results are shown with $k = 2$. The resulting distribution of *rule structures* and data instances is provided in Table 5.6

5.2.3.2 Post-filtering for coverage

The results of base test of simulated annealing rule learning method is given in Table 5.7. The legend of column headings is given in Table 5.8.

This base test shows the results of simulated annealing with no additional improvements. Root lexemes of words are not included in rules. Zero errors are not disallowed while training. Parent node appending is not applied.

Two lines of results are reported. The first line contains results with no post-filtering of the learned rules. All rules that cover at least a single training instance are included in predicting the test set. The second line contains results when learned rules covering less than 2 training set instances are removed. This post filtering operation drops the recall rate with only 3 points.

As shown in recall rates, nearly 23 to 26 percent of the positive test instances cannot be extracted.

Examining number of false positives, effects of different choices for defining error can be seen more clearly. Relaxation of *closed-world assumption* to *closed-concepts* halves the number of false positives. Half of the initially returned false positives share no words in common with tagged examples.

Further relaxing this assumption to *closed-relations*, the number of false positives are nearly halved again. This means that only one quarter of the returned non-matching relations share only one concept with tagged relations.

The last quarter of non-matching returned relations may be because of partial concept matches.

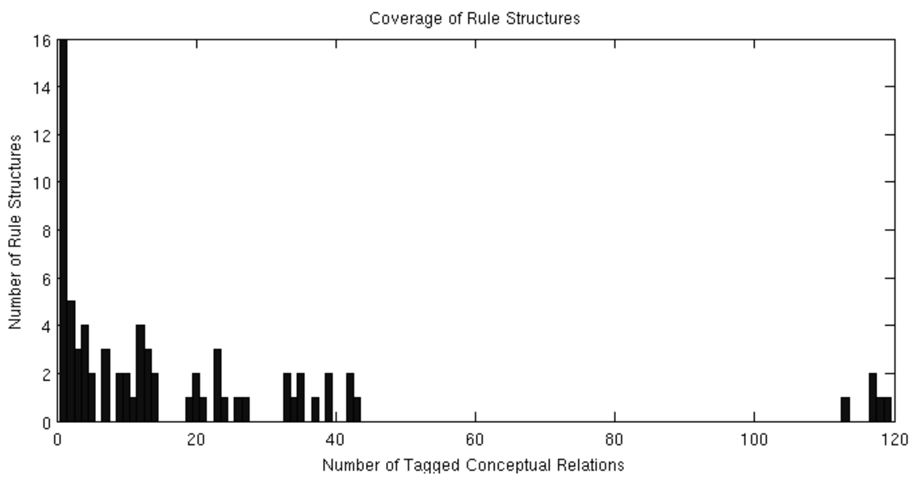


Figure 5.1: Histogram of data instance coverage of *rule structures*

Table 5.6: Tagged instances and Rule structures after filtering

Name	Instances	
	k = 2	k = 10
Tagged instances	528	447
location	151	112
modifier	268	249
locative modifier	84	69
measurement	25	17
Unique *rule structures*	43	16
location	16	3
modifier	17	9
locative modifier	7	3
measurement	3	1

Even if relaxing error definition halves the number of false positives, precision percentages do not get doubled. This is because of the relation of precision with true positives as well as false positives.

Table 5.7: Data-driven method: Base test

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	*CCA*	*CRA*		*CWA*	*CCA*	*CRA*
Test-1	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%

Table 5.8: Legend for experiment reports

Legend	
L	Inclusion of root lexeme (Y if included)
A	Appending of additional parent node (Y if appended)
0	Disallowing of 0-errors (Y if disallowed)
S	Inclusion of suffixes of root lexeme (Y if included)
MC	Minimum coverage for rules
TP	True positives
FN	False negatives
FP	False positives
CWA	Error measured with closed-world assumption
CCA	Error measured with closed-concepts assumption
CRA	Error measured with closed-relations assumption

5.2.3.3 Inclusion of root lexeme

The low precision in the previous test may be improved by including root lexemes in rule constraints. This would enhance the algorithm's awareness of the matching context. The results are compared with the previous test in Table 5.9.

When the root lexemes are included in the rules, precision increases by at least ten percent in all cases. Number of false positives in all cases dropped significantly (nearly to 1/3) compared to previous test.

A disadvantage is the decreased generalization performance of the trained model. The number of true positives are nearly halved, and recall values drop by 26% and 31% respectively for minimum cover settings of 1 and 2.

The difference in drop in recall values can be explained by effect of minimum cover requirement. When minimum cover is 2, all rules matching to only a single training instance are dropped. As the model overfits to the training set, this causes more decreasing of recall value at *mincover* = 2 compared to *mincover* = 1.

Table 5.9: Data-driven method: Inclusion of root lexeme

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	*CCA*	*CRA*		*CWA*	*CCA*	*CRA*
Test-1	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%
Test-2	Y N N N	1	267	261	315	179	93	50.6%	45.9%	59.9%	74.2%
		2	223	305	195	108	58	42.2%	53.3%	67.4%	79.4%

5.2.3.4 Disallowing zero-errors

Inclusion of root lexemes causes the learned rules to overfit the training set. The reason can be found in the implemented simulated annealing algorithm. A candidate rule reaches zero error when a lexeme constraint is included, and this candidate is assigned to *best rule*. The algorithm will not be able to select a rule with no lexeme constraint because the current *best rule* already has the best possible error. Algorithm's preference for more generalized rules makes the matter worse, by removing every morphological constraint and leaving only the root lexemes as the rule's constraint.

This problem may be solved by disallowing exploration of candidate rules that return an error of zero. This would prevent the algorithm from reaching the global minima of the error function. The algorithm will never include only the root lexeme constraint, and will have to keep morphological feature constraints in the learned rule in order to compensate for the error.

The results of this test appears in Table 5.10, compared with the previous tests.

The recall rates are 3 points higher than the first test, which renders inclusion of root lexemes combined with disallowing of zero-errors better generalizing to unseen examples. Comparing number of false positives, the third test has nearly the same error in the most relaxed error definiton. False positives decreased slightly in the strictest error definition.

Compared with the second test, the high gain in precision is almost lost. The precision was gained with overfitting.

Table 5.10: Data-driven method: Disallowing zero errors

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	CCA	CRA		CWA	CCA	CRA
Test-1	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%
Test-2	Y N N N	1	267	261	315	179	93	50.6%	45.9%	59.9%	74.2%
		2	223	305	195	108	58	42.2%	53.3%	67.4%	79.4%
Test-3	Y N Y N	1	419	109	870	441	245	79.4%	32.5%	48.7%	63.1%
		2	406	122	729	359	219	76.9%	35.8%	53.1%	65.0%

5.2.3.5 Appending one additional parent node to rule representation

The low precision of base test may be solved with other methods than inclusion of root lexeme. If the low precision is caused by smaller rules (in number of nodes) matching to sentences instead of larger rules, addition of an additional parent node may help the algorithm to differentiate between different sentences.

The results of this test, compared with the base test, is shown in Table 5.11. Compared with the base test, the precision is increased by a few percent, with a corresponding drop in recall.

Table 5.11: Data-driven method: Appending of an additional node

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	CCA	CRA		CWA	CCA	CRA
Test-1	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%
Test-4	N Y N N	1	361	167	710	366	183	68.4%	33.7%	49.7%	66.4%
		2	401	127	707	360	210	75.9%	36.2%	52.7%	65.6%

5.2.3.6 Inclusion of string suffixes of root lexeme

Another alternative to including the root lexeme in the rule is to only include string suffixes of it. As most medical terms are derived from counterparts in Western languages, last two or three letters of lexemes might convey hints on the role of the word.

Analysis of the number of unique string suffixes of two and three letters is given as a histogram in Figure 5.2. Utilizing this histogram, suffixes that occur in at least 10 words are included in the features. The filtered suffixes and their number of occurences is shown in Table 5.12.

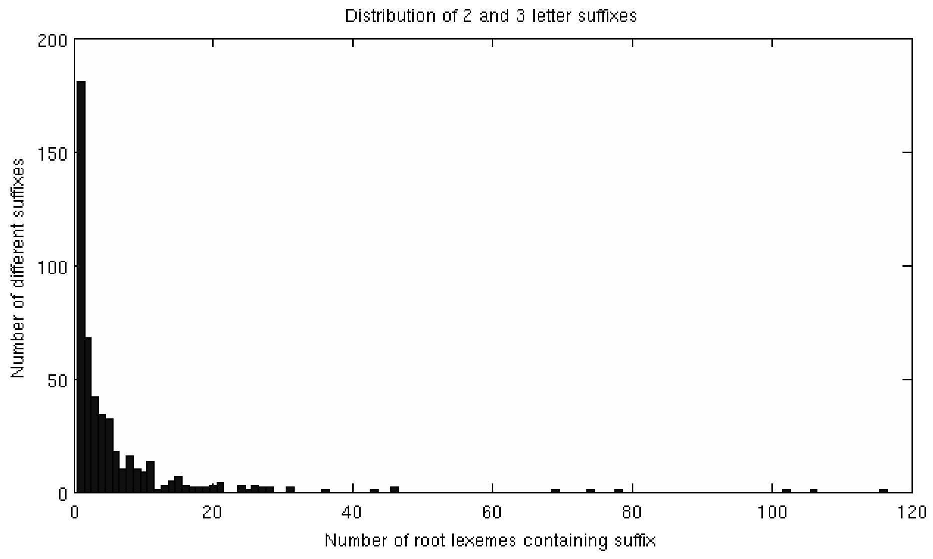


Figure 5.2: Distribution of 2 and 3 letter suffixes of root lexemes

The suffixes are included in the morphological tags section of the rules. Each suffix can be constrained to exist or be absent like morphological tags. Inclusion of suffixes increased the number of features from around 50 per node to 125 per node in rule representation. Doubling in branching factor could be compansated with an increase of number of iterations. But, a linear increase in number of iterations with the branching factor would not have enough effect. In each step of the algorithm only a single feature is changed, but the number of combinations increases exponentially with the branching factor. Instead, the exploration constant in the algorithm is increased. Instead of exploring 5 neighbors in every step, the algorithm explores 20 neighbors to compensate for the increase in branching factor.

The results of this test is provided in Table 5.13 compared with the base test. The recall is dropped by 3 to 5 points, with a corresponding increase of 3 to 13 points in precision. The increase in precision is maximum at strictest definition of error. Although inclusion of string suffixes caused some loss of generalization, it is not fatal compared to previous tests. Considering the high increase in precision, slight decrease of generalization performance and

Table 5.12: Counts of suffixes with more than 9 instances

Suffix	Instances	Suffix	Instances	Suffix	Instances
-ak	36	-im	15	-sol	27
-al	106	-in	16	-st	14
-an	14	-ir	10	-ta	26
-ap	26	-it	10	-te	11
-ar	20	-ite	10	-ter	21
-ağ	17	-ki	24	-tik	28
-aş	18	-kik	18	-tle	26
-bt	11	-lak	11	-tur	10
-bu	13	-le	116	-um	20
-cak	10	-lın	10	-ur	10
-cm	21	-mal	31	-us	15
-de	11	-men	10	-ut	21
-ek	27	-mm	11	-var	15
-em	11	-nal	25	-ve	46
-en	28	-nt	16	-yon	43
-ent	14	-ol	74	-yum	14
-er	102	-on	46	-yut	15
-eri	11	-pta	12	-yük	19
-et	15	-ra	21	-zle	69
-fra	15	-ral	16	-çap	17
-gin	15	-re	13	-ük	19
-her	11	-rek	24	-üm	11
-ik	78	-ri	11	-ğer	31
-iki	11	-rik	13	-ın	24
-il	11	-rt	11	-ır	10
-ile	20	-sağ	14		

recall rate is within acceptable bounds.

Table 5.13: Data-driven method: Inclusion of lexeme suffixes

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	*CCA*	*CRA*		*CWA*	*CCA*	*CRA*
Test-1	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%
Test-5	N N N Y	1	380	148	569	317	171	72.0%	40.0%	54.5%	69.0%
		2	360	168	410	233	126	68.2%	46.8%	60.7%	74.1%

5.2.3.7 Support Vector Machines

In order to conduct support vector machine learning, the dataset is mapped to vector space. By the definition of the mapping method used, separate *rule structures* require separate support vector machines. Also, each single data instance is mapped to multiple vectors in the vector space. The details of the mapping can be found in Chapter 4 with a discussion on possible problems.

The number of positive instances in conceptual relation space per *rule structure*, and number of positive and negative instances in vector space per support vector machine is given in Table 5.14. In the same table number of nodes in the *rule structure*, and corresponding number of features in vector space is provided.

Number of features per SVM grows linearly with number of nodes in *rule structures*. Each node has a fixed list of features, and they are appended together in vector space. On the other hand, number of instances per SVM grows exponentially with number of nodes in *rule structures*. A *rule structure* is basically an undirected acyclic graph, and all possible alignments of this graph with the input sentences are included in the vector space as distinct instances.

Deciding with multiple instances Each possible conceptial relation is represented by multiple instances in vector space. A decision problem occurs when the trained SVM model disagrees on the labels of different instances of the same relation.

Comparing the situation with the rule learning algorithm, a conceptual relation's existance can be decided if only a single instance is classified as positive, ignoring negative classifications.

Table 5.14: Vector space representation

Title	Relation space		Vector space		
	Nodes	Pos.Instances	Features	Pos.Instances	Neg.Instances
SVM-1	2	113	114	113	2961
SVM-2	5	13	318	38	12659
SVM-3	5	39	318	143	12554
SVM-4	3	19	182	19	2201
SVM-5	4	117	250	429	14283
SVM-6	3	33	182	33	1935
SVM-7	2	37	114	37	3037
SVM-8	3	12	182	12	1956
SVM-9	8	27	522	682	138833
SVM-10	8	26	522	630	109853
SVM-11	7	24	454	594	77355
SVM-12	3	13	182	13	2207
SVM-13	4	14	250	14	2640
SVM-14	4	34	250	35	3173
SVM-15	4	20	250	20	3050
SVM-16	3	11	182	11	4104
SVM-17	3	33	182	33	1970
SVM-18	5	23	318	107	13459
SVM-19	5	39	318	145	13130
SVM-20	5	12	318	21	4277
SVM-21	4	42	250	166	14546
SVM-22	5	12	318	70	24831
SVM-23	4	35	250	35	4242
SVM-24	3	12	182	12	1991
SVM-25	5	23	318	107	13168
SVM-26	4	35	250	35	4445
SVM-27	4	117	250	443	14148
SVM-28	3	13	182	13	1990
SVM-29	8	43	522	4333	462578
SVM-30	6	118	386	2035	77724
SVM-31	4	42	250	166	14425
SVM-32	6	119	386	2104	76395
SVM-33	5	14	318	74	13201
SVM-34	5	10	318	10	1877
SVM-35	4	21	250	21	4717
SVM-36	5	23	318	24	6837
SVM-37	4	20	250	21	3571
SVM-38	5	10	318	11	2379
SVM-39	3	10	182	31	2236
SVM-40	5	8	318	11	2379
SVM-41	4	11	250	11	2104
SVM-42	4	24	250	71	11151
SVM-43	3	31	182	31	2185

In rule learning, even if a single alignment of a rule matches with the sentence, that aligned conceptual relation in the sentence is extracted.

Another alternative is to exploit the vector space representation. Every possible alignment of a *rule structure* with the candidate conceptual relation exists in the vector space, and all of them are already classified. A candidate conceptual relation may be returned only when number of positive classifications are greater than number of negative classifications of its instances in vector space. Using this weighted scheme caused no differences in the experiments.

In order to calculate matches in relation space, each test sentence's all possible alignments with *rule structures* are fed into corresponding trained support vector models. The extracted relations by all support vector machines are merged. Performance calculations are performed on extracted unique relations per sentence.

Parameter tuning A linear binary SVM classifer has two parameters, C, the penalty parameter, and w_1, weight of penalties to positive instances compared to negative instances.

Setting $w_1 = 1$ has the effect of considering both classes with the same importance. In this work, w_1 is assigned as the inverse of ratio of positive instances to negative instances. Thus, the positive and negative instances that lie on the wrong side of the separation plane are given a penalty proportional to their prior probabilities. Note that the prior probabilities are calculated on the vector space, not relation space.

For the penalty parameter C, a tuning procedure is applied. For every training set, the set is divided into 5 random folds, and 5-fold cross validation is performed with different values of C (between 2^{-5} and 2^{15}). Note that this cross validation is independent of the cross validation procedure for measuring test performances; this operation is performed only on the training sets. The target for cross validation is highest F1-score classification performance in vector space.

As a result of tuning process, independent w_1 and C values are chosen for every training set of every SVM.

Experimental results Two experiments are performed with support vector machines. The first experiment tests the performance with no string suffixes. In the second experiment, string suffixes for each node are appended to input vectors as separate features.

The comparison of SVM tests are given in Table 5.15. Inclusion of string suffixes drops recall of second experiment by 10 points. This is due to overfitting to data. Still, the absolute change in number of true positives is low. Inclusion of string suffixes again increased precision significantly.

Table 5.15: SVM extraction performance

Title	S	TP	FN	FP			Recall	Precision		
				CWA	*CCA*	*CRA*		*CWA*	*CCA*	*CRA*
Test-6	N	221	307	768	582	210	41.9%	22.3%	27.5%	51.3%
Test-7	Y	165	363	299	213	67	31.3%	35.6%	43.7%	71.1%

5.2.4 Comparison of Extraction Results

Results of experiments of rule based, simulated annealing based data-driven, and SVM based data-driven algorithms (with filter threshold $k = 2$) are given in a single Table 5.16 for comparison. Also, results of the same tests for filter threshold $k = 10$ is provided in Table 5.17.

Increasing filtering threshold resulted in better recall rates (around 10 points) in simulated annealing algorithm. Precision is not affected at all in simulated annealing algorithm. The most drastic effect can be seen in change of SVM performance, where increasing filtering threshold caused doubling of recall rates, and 10 to 20 points of increase in precision.

5.2.5 Effects of Morphological Disambiguation and Dependency Parsing

The results reported for data-driven method in the previous experiments are based on manually tagged morphological disambiguation and dependency parsing data. The results were reported without a morphological disambiguation and dependency parsing step in order to show the performance of the pure algorithm. In a real-life scenario, the data-driven algorithms will perform after morphological disambiguation and dependency parsing steps.

Table 5.16: Comparison of extraction results (filtering $k = 2$)

Title	L A 0 S	MC	TP	FN	FP *CWA CCA CRA*	Recall	Precision *CWA CCA CRA*
Rule based							
Test-0			137	289	752 539 285	32.2%	15.4% 20.3% 32.5%
Data-driven (Simulated Annealing)							
Test-1	N N N N	1	404	124	873 388 231	76.5%	31.6% 51.0% 63.6%
		2	389	139	781 361 210	73.7%	33.2% 51.9% 64.9%
Test-2	Y N N N	1	267	261	315 179 93	50.6%	45.9% 59.9% 74.2%
		2	223	305	195 108 58	42.2%	53.3% 67.4% 79.4%
Test-3	Y N Y N	1	419	109	870 441 245	79.4%	32.5% 48.7% 63.1%
		2	406	122	729 359 219	76.9%	35.8% 53.1% 65.0%
Test-4	N Y N N	1	361	167	710 366 183	68.4%	33.7% 49.7% 66.4%
		2	401	127	707 360 210	75.9%	36.2% 52.7% 65.6%
Test-5	N N N Y	1	380	148	569 317 171	72.0%	40.0% 54.5% 69.0%
		2	360	168	410 233 126	68.2%	46.8% 60.7% 74.1%
Data-driven (Support Vector Machines)							
Test-6	N		221	307	768 582 210	41.9%	22.3% 27.5% 51.3%
Test-7	Y		165	363	299 213 67	31.3%	35.6% 43.7% 71.1%

Table 5.17: Comparison of extraction results (filtering $k = 10$)

Title	L A 0 S	MC	TP	FN	FP *CWA*	FP *CCA*	FP *CRA*	Recall	Precision *CWA*	Precision *CCA*	Precision *CRA*
Data-driven (Simulated Annealing)											
Test-1	N N N N	1	395	52	792	616	488	88.4%	33.3%	39.1%	44.7%
		2	394	53	746	587	478	88.1%	34.6%	40.2%	45.2%
Test-2	Y N N N	1	303	144	162	128	88	67.8%	65.2%	70.3%	77.5%
		2	259	188	134	105	73	57.9%	65.9%	71.2%	78.0%
Test-3	Y N Y N	1	397	50	548	415	289	88.8%	42.0%	48.9%	57.9%
		2	395	52	495	379	278	88.4%	44.4%	51.0%	58.7%
Test-4	N Y N N	1	370	77	653	503	374	82.8%	36.2%	42.4%	49.7%
		2	361	86	577	451	348	80.8%	38.5%	44.5%	50.9%
Test-5	N N N Y	1	370	77	306	221	138	82.8%	54.7%	62.6%	72.8%
		2	362	85	274	201	128	81.0%	56.9%	64.3%	73.9%
Data-driven (Support Vector Machines)											
Test-6	N		332	110	980	713	195	75.1%	25.3%	31.8%	63.0%
Test-7	Y		241	155	162	117	21	60.9%	59.8%	67.3%	92.0%

Due to low number of instances in our dataset, satisfactory results with Yuret and Türe's disambiguator [41] could not be achieved.

Before analyzing effects of automated dependency parsing to the algorithms, a stand-alone test of dependency parser is made. Table 5.18 shows the results of dependency parsing on our dataset. This test is performed with leave one out cross validation ($k = numberofexamples$). The results of Eryigit et al. [11] on Turkish Treebank [29] is also shown in the table for comparison.

Table 5.18: Performance of Dependency Parsing

	AS_L	WW_L	AS_U	WW_U
Abdominal radiology reports dataset	68.3%	71.9%	75.8%	80.3%
Turkish Treebank, CoNNL-X (Eryiğit et. al.)	67.0%	-	76.0%	82.7%

AS_L is the labeled attachment score; the percentage of inflectional groups that are attached to the correct inflectional group of their head word with the correct dependency label.

AS_U is the unlabeled attachment score; the percentage of inflectional groups that are attached to the correct inflectional group of their head word, with no constraints on dependency label.

WW_L is the labeled word-to-word attachment score; the percentage of inflectional groups that are attached to any inflectional group of their head word with the correct dependency label. Value of this score for Turkish Treebank was not reported.

WW_U is the unlabeled word-to-word attachment score; the percentage of inflectional groups that are attached to any inflectional group of their head word, with no constraints on dependency label.

The results given for Eryigit et. al. is the results of IG-based dependency parsing on gold standard mophological analyses (absent of automatic disambiguation process).

Although our dataset is much smaller in size compared to Turkish Treebank, comparable results are achieved. The reason might be the highly specialized contents of the dataset.

In this work, word-to-word labeled attachments are used as inputs for proposed data driven algorithms.

A copy of the first data-driven test is performed for measuring the effects of automated dependency parsing. Training for data-driven algorithm is made with the manually tagged dependency links. But for the test sets, instead of using the manually tagged dependency links, output of dependency parser is used. The dependency parser is trained with the same training set used to train data-driven algorithm. 10-fold cross validation is performed. The results are shown in Table 5.19.

The inclusion of dependency parsing in the process drops recall rates by 13 points, and precision rates by 8 points in average.

Table 5.19: Effects of dependency parsing on base test

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	*CCA*	*CRA*		*CWA*	*CCA*	*CRA*
Test-1 w/o dep.	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%
Test-1 w/ dep.	N N N N	1	332	196	1040	550	263	62.9%	24.2%	37.6%	55.8%
		2	318	210	834	452	244	60.2%	27.6%	41.3%	56.6%

5.2.6 Evaluation by Domain Experts

All the extraction results reported so far are obtained by cross validation on a single dataset. In order to obtain objective evaluation by medical doctors, a third test dataset is created. This dataset consists of 7 abdominal reports, 34 sentences, and 117 tagged conceptual relations. Two domain experts tagged the dataset with consideration of the desired output representation.

A rule-learner simulated annealing model is trained with the second dataset which was used in previous experiments, and this model is applied on the mentioned third dataset.

Only the base model (with no variations on the algorithm) is tested in order to compare its results with the results of base model on cross validation of second dataset. The results are provided in Table 5.20. Note that as the test datasets differ, the reported recall and precision values cannot be directly compared. Rather, the reported difference in measurements should be applied to the overall results of Table 5.16, to understand the possible effects of variations of the algorithm on this expert-tagged dataset.

Minimum cover parameter did not affect the results significantly. A recall of 65.8% is achieved. The precision is 37.0% under closed world assumption and 65.3% in the most relaxed definition of error.

Regarding the low values of recall and precision, an important consideration is lack of filtering for the third dataset. The second dataset was filtered for insufficient number of training examples prior to running the experiments. This filtering was done on the entire dataset, not only the partition used for training, in order to report the performance more accurately. In the third dataset, no filtering could be performed. As tagging quality could not be compansated by filtering, we attribute some of the loss in performance to tagging quality.

Table 5.20: Data-driven method: Third dataset (Expert opinion)

Title	L A 0 S	MC	TP	FN	FP			Recall	Precision		
					CWA	*CCA*	*CRA*		*CWA*	*CCA*	*CRA*
Test-1 *cross val.*	N N N N	1	404	124	873	388	231	76.5%	31.6%	51.0%	63.6%
		2	389	139	781	361	210	73.7%	33.2%	51.9%	64.9%
Test-1 *expert dataset*	N N N N	1	77	40	131	66	41	65.8%	37.0%	53.8%	65.3%
		2	71	46	115	61	35	60.7%	38.2%	53.8%	67.0%

5.3 Discussion

One rule based and one data driven algorithm are proposed for information extraction. The rule based algorithm is also tested on a retrieval dataset. A baseline word search algorithm is applied to retrieval dataset for comparison.

A rule learning model is proposed for data driven extraction of medical relations. Structure of the rules limited the ability of the algorithms by dividing the already small dataset into smaller orthogonal groups.

Two alternate implementations of the data driven algorithm is tested. One is based on learning rule constraints with simulated annealing. The other is based on mapping the data to vector space, and classifying in the vector space with linear support vector machines.

Simulated Annealing vs. SVM Classification

Although support vector machines are excellent classifiers, they were unable to perform as well as the simulated annealing learning algorithm. The reason for this inability should be sought in mapping from graph based rule model to vector space. With better mapping methods, or suitable graph-based kernels, support vector machines would provide better results. Experimenting with non-linear support vector machines might also provide better results, but with a counterbalancing requirement of more computational time.

Variations of methods

Variations of features for rule structures and variations on algorithm are compared. Their effects are analyzed.

- Inclusion of root lexemes in rule representation yielded better precision, but caused the algorithm to overfit to the training data, decreasing recall on test set. Overfitting caused by root lexemes is prevented by disallowing learning of zero-error yielding rules during the training process.
- As an alternative to including root lexemes, string suffixes of root lexemes are appended to the rule structures. This variation improved precision results drastacially

in the strictest measure of error, almost comparable to inclusion of roots' lexemes.

Error measurement

Three different definitions of error is provided for evaluating false positives. All experiments are reported with true positives, false negatives, and false positives based on these three definitions. One recall and three alternative precision values are also calculated. Based on the assumptions on the tagged training set, any of the error metrics might be considered as the actual error. Reporting error under a variety of possible definitions is chosen in order to show the reader the sources of low precision under closed world assumption.

Effects of preliminary steps

Dependency parsing performance on the extraction dataset is compared with the previous work on Turkish Treebank. No significant loss is seen in spite of the relatively small size of the extraction dataset. It is believed that this absence of loss is caused by the small breadth of the dataset which balances the effects of the dataset size.

The effects of dependency parsing on proposed algorithms' results are shown separately. Only a single comparison is made with the base data-driven experiment to show the effects and loss in performance. Similar loss would apply to other experiments as well, when preceded by automated dependency parsing.

Automated morphological disambiguation could not be performed or included in the measurements due to relatively small size of dataset. The disambiguator failed to learn proper rules in the absence of enough data.

Expert Evaluation

A third dataset, similar to the extraction dataset, is tagged by medical doctors. This dataset is tagged with considering the target representation. A base data-driven model is trained with the extraction dataset, and is applied on the expert-tagged dataset. A recall of 65.8% and a precision of 37.0% to 64.3% (based on definition of error) is achieved. As no proper filtering could be applied to the third dataset, the results are lower than expected. This is attributed to tagging quality. Better results would be achieved in a larger dataset, and by utilizing the proposed variations on the algorithm.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Existing non-English medical NLP systems use either manually constructed (but limited) ontologies, or machine translation methods. Considering the size of established medical ontologies in English, manually constructed ontologies and lexicons for a single work limit the scope of retrieval and extraction systems to prototypes. On the other hand, using translation techniques to map non-English text (either as a whole or on a phrase basis) to a proven ontology like SNOMED-CT would have significant benefits, but is prone to accumulating errors at each step regarding the translation.

In this work, instead of automatic translation approaches or manually constructed ontologies, a different but unique approach is preferred. We focused on extracting the relations between different medical phrases in the text. Use of a medical lexicon and ontology is left out delibarely, in order to see the baseline the Turkish language can provide for this task in isolation of other factors.

This thesis is one part of a larger project on information retrieval from Turkish radiology reports. Other two parts of the project focus on medical lexicons, Turkish to English translation (for locating terms in ontology), and usage of SNOMED-CT ontological relations. On the contrary, the part that was implemented by this thesis relies only on analysis of Turkish sentence structures to detect medical relations within text.

The starting point of this thesis lies in the observation that a person with no knowledge except for Turkish language grammar may still understand Turkish radiology reports, even if partially. The limits of this observation is explored throughout the thesis. We proposed one rule based and one data driven algorithm. We tested the rule based algorithm in a retrieval context, and both algorithms in an extraction context.

In retrieval context, rule based algorithm is compared to a baseline algorithm that checks for existance of query words. This baseline algorithm achieved very high recall and very low precision rates, which shows that most of the documents in question contains the query words. Although the rule based algorithm provided higher precision, it resulted in lower recall rates. In the end, it did not achieve the desired performance. This may be explained with the breadth of the test set which contained different kinds of radiology reports, and errors related with manually constructing the rules.

In extraction context, rule based algorithm is compared with different versions of the data driven algorithm. Extraction dataset consisted of only abdominal radiology reports, in order to provide sufficient samples for training the data driven algorithm.

There is no single error measure in an extraction scenario. In order to measure error in extraction dataset, 3 different measures of false positives are calculated. False positives are calculated under *closed world* assumption, a relaxed *closed world on concepts* assumption, and a more relaxed *closed world on relations* assumption. These three alternatives can be seen as three different definitions of error, depending on the aims of a study. For every experiment, one recall, and three alternative precision values were reported. Our precision results were poorer than expected in the strictest definition of error (*closed world* assumption). We attribute this to the tagging quality of the input, a *closed world* assumption was too strict for our dataset.

Tagged relations in the extraction dataset consisted of only a subset of the rule based algorithm's output relation types. Thus only a subset of the rule based algorithm's output could be compared, ignoring its intermediate relations. This incompability resulted in a drop in its expected performance. The rule based method could only achieve 32.2% recall, and 15.4% to 32.5% precision. The main cause of this low success lies in incorrect extraction of phrase boundaries. It is more suited to information retrieval than precise extraction.

Data driven method is less prone to human error than the rule based, as it is based on training with target representation. We reported performance of the data driven method with k fold cross validation, with $k = 10$.

Initially, we implemented a covering algorithm. In each step, an uncovered training example is selected as seed. A rule that covers maximum number of training examples and results in minimum error is generated based on the seed. For generation of the rule, a variation of simulated annealing algorithm is used. Error function is chosen as number of correct matches vs. all matches. Ties are broken with a generalization metric on the candidate rules, favoring more generalized rules. A post-filtering operation on minimum cover was employed, removing rules covering less than *mc* training instances; in order to make sure learned rules would generalize to test data. $mc = 1$ and $mc = 2$ were reported in the experiments.

The learned rule structures are based on subtrees of the dependency graph of the input sentence, with dependency directions, dependency labels, lexeme and morphological features being optional. Each rule structure holds nearly 50 binary features per node. Besides high number of features, another disadvantage of the chosen rule structure is that it partitions the dataset into orthogonal groups. By definition, a rule structure that matches a relation between n words can not match a relation between $m \neq n$ words.

Simulated annealing algorithm changes only a single feature in each iteration. Combined with high number of features, this approach made the defined error function mostly flat, thus not being capable of differentiating between successive rule candidates. In order to overcome this problem, we modified the simulated annealing algorithm to explore multiple neighbors at a time. This resulted in better convergence.

Another alternative to covering algorithm was to map the rule structures to vector space, where proven methods could be used for prediction. Linear support vector machines were used for this purpose. With this approach, a single training example in rule structure space might be mapped to multiple instances in vector space. This caused exponential increase in the number of training examples in vector space. Also, as the proposed rule structures were orthogonal, multiple SVM models had to be trained. We preferred linear SVMs due to computational time requirements.

For the covering algorithm, our base case achieved 76.5% recall and 31.6% to 63.6% precision. In order to compansate for low precision, we tried including root lexeme of words in the rule structure. This resulted in an increase of 13% in precision, but with a corresponding drop of 26% in recall. We attributed the drop in recall to overfitting, caused by root lexemes. We ran another test with a limitation on error function so that zero-error yielding rules would be prevented in learning step. This limitation increased recall rates back to 79.4%, but also lost the gains in precision, resulting in only around 2% of precision increase compared with the base case.

As another alternative to using root lexemes, we tried appending 2 and 3 letter suffixes of root lexemes to the morphological tags. The intuition behind this test was simple; most of the medical terms in the reports were directly borrowed from western languages, and most of the time last letters of these words conveyed their role. We believed this would not cause the overfitting problem, but still keeping the advantages of having root lexeme in rule structure. This modification resulted in nearly 10% increase in precision values, with a corresponding drop of 5% in recall. Although not as successful as using the root lexeme itself, this modification was proved to be useful in increasing precision.

Vector space experiments resulted in much worse results in both recall and precision compared to covering algorithm base experiment. We attribute this to the high number of SVM models to be trained, with a very small number of positive training instances per SVM. The SVM models could not be trained properly due to insufficient training data, resulting in bad performance (41.9% recall and 22.3% to 51.3% precision). We tried adding 2 and 3 letter suffixes of root lexemes to the training vectors, which resulted in an increase of 13% to 20% increase in precision, followed by a drop of 10% in recall. Although suffixes once again increased precision, they caused overfitting, thus decreasing recall.

We performed most of our tests with manually tagged morphological disambiguations and dependency graphs. But our intention was to train a morphological disambiguator and a dependency parser using the training set, and conduct the experiments by automated disambiguation and dependency parsing. Due to the small size of our dataset, we could not succeed in training the disambiguator. We ran an experiment only with automated dependency parsing, and compared the result with the covering algorithm base experiment. Using dependency output of a data-driven parser resulted in a drop of 13% in recall and 6% in precision. We were expecting higher drop rates. In order to interpret the results with a better perspective, we tested the data-driven dependency parser in our dataset, and compared its results with the previous dependency parsing results on Turkish treebank. We got comparable percentages from this experiment.

In order to obtain expert opinion, a smaller dataset similar to the extraction dataset is tagged by two medical doctors, with considering the desired output representation. A rule-learner covering algorithm model is trained on the extraction dataset, and this third dataset is used as a test dataset. We achieved 65.8% recall and 37.0% to 65.3% precision in this expert-tagged dataset. Note that these percentages are not comparable with the previous percentages reported, as the test datasets differ. No filtering could be applied on the test dataset, and the training was performed with the base case of rule learning algorithm. With a larger test dataset that would not require filtering, and with use of proposed variations on the algorithm, these percentages would be higher.

This thesis proposed that, even in the lack of medical lexicons and ontologies, structured information from Turkish radiology reports could be properly extracted. Although achieved recall and precision rates are lower than our initial expectations, they show that extraction is possible without any medical lexicon or ontology. Still, a medical lexicon or ontology would increase these results significantly.

This work is not intended to be a standalone medical information extraction system on its own. Rather, the baseline performance that Turkish language provides in absence of any medical information is measured. The proposed methods map free text inputs to a structured output, which might be more easily filtered and post-processed to be used in a medical information extraction system by the use of domain information.

6.2 Future Work

Although it is stated that this work is not intended to be a standalone medical information extraction system, it may be used as a component of such a system for Turkish radiology reports.

The proposed algorithms would perform better when preceded by a medical lexical tagger. Semantic tagging of input words would improve the precision compared to usage of only POS and morphological tags. An entity recogniser might also be employed so that the algorithm will not have to deal with phrases consisting of multiple words. These measures would reduce error rates.

The output of the algorithm may be further filtered for meaningful relations. Given dependency links of a sentence, the proposed algorithms handle extracting of probable medical relations. After the extraction of the relations, relations containing unmeaningful concepts can be easily filtered out by domain knowledge.

REFERENCES

[1] Ahmet Afsin Akin and Mehmet Dundar Akin. Zemberek, an open source NLP framework for Turkic Languages. 2007.

[2] Sabine Buchholz and Erwin Marsi. Conll-x shared task on multilingual dependency parsing. In *In Proc. of CoNLL*, pages 149–164, 2006.

[3] André Coutinho Castilla, Sérgio Shiguemi Furuie, and Eneida A. Mendonça. Multilingual information retrieval in thoracic radiology: Feasibility study. In Klaus A. Kuhn, James R. Warren, and Tze-Yun Leong, editors, *MedInfo*, volume 129 of *Studies in Health Technology and Informatics*, pages 387–391. IOS Press, 2007.

[4] Çağrı Çöltekin. A freely available morphological analyzer for turkish. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

[5] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer Publishing Company, Incorporated, 2008.

[6] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In *Proceedings of the 17th international joint conference on Artificial intelligence - Volume 2*, pages 1251–1256, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[7] Sean P. Engelson and Ido Dagan. Sample selection in natural language learning. In *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 230–245, London, UK, 1996. Springer-Verlag.

[8] Gülşen Eryiğit and Eşref Adalı. An affix stripping morphological analyzer for Turkish. In *Proceedings of the International Conference on Artificial Intelligence and Applications*, pages 299–304, Innsbruck, 16-18 February 2004.

[9] Gülşen Eryiğit, Joakim Nivre, and Kemal Oflazer. The incremental use of morphological information and lexicalization in data-driven dependency parsing. In *In Proceedings of the 21st International Conference on the Computer Processing of Oriental Languages, Sentosa*, 2006.

[10] Gülşen Eryiğit and Kemal Oflazer. Statistical dependency parsing of Turkish. In *Proceedings of the 11th EACL*, pages 89–96, Trento, 3-7 April 2006.

[11] Gülşen Eryiğit, Joakim Nivre, and Kemal Oflazer. Dependency parsing of Turkish. *Comput. Linguist.*, 34:357–389, September 2008.

[12] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[13] C Friedman, Po Alderson, Jh Austin, Jj Cimino, and Sb Johnson. A general natural-language text processor for clinical radiology. 1994.

[14] C. Friedman and G. Hripcsak. Evaluating natural language processors in the clinical domain. *Methods of information in medicine*, 37:334–44, 1998.

[15] C. Friedman and G. Hripcsak. Natural language processing and its future in medicine. *Acad Med*, 74(8):890–895, August 1999.

[16] Carol Friedman, George Hripcsak, Lyuda Shagina, and Hongfang Liu. Representing information in patient reports using natural language processing and the extensible markup language. *Journal of the American Medical Informatics Association*, 6(1), Jan/Feb 1999.

[17] Dilek Hakkani-Tür, Kemal Oflazer, and Gökhan Tür. Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36:381–410, 2002. 10.1023/A:1020271707826.

[18] George Hripcsak, Gilad J Kuperman, Carol Friedman, and Daniel F Heitjan. A reliability study for evaluating information extraction from radiology reports. *Journal of the American Medical Informatics Association*, 6:143–150, 1999. 10.1136/jamia.1999.0060143.

[19] Scott Huffman. Learning information extraction patterns from examples. In *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260. Springer, 1995.

[20] Kimmo Koskenniemi. A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*, ACL '84, pages 178–181, Stroudsburg, PA, USA, 1984. Association for Computational Linguistics.

[21] Agnieszka Mykowiecka, Małgorzata Marciniak, and Anna Kupść. Rule-based information extraction from patients' clinical data. *Journal of Biomedical Informatics*, 42(5):923–936, October 2009.

[22] Volker Nannen. An affix stripping morphological analyzer for Turkish. Master's thesis, Istanbul Technical University, Istanbul, 2002.

[23] Joakim Nivre. Dependency grammar and dependency parsing. Technical report, Växjö University, 2005.

[24] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Stetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering Journal*, 13(2):99–135, 2007.

[25] Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Stetoslav Marinov. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York, NY, 8-9 June 2006.

[26] Kemal Oflazer. Two-level description of Turkish morphology. In *Proceedings of the sixth conference on European chapter of the Association for Computational Linguistics*, EACL '93, pages 472–472, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics.

[27] Kemal Oflazer, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. Corpus annotation for parser evaluation. In *In Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC*, pages 35–41, 1999.

[28] Kemal Oflazer and Ìlker Kuruöz. Tagging and morphological disambiguation of turkish text. In *Proceedings of the fourth conference on Applied natural language processing*, ANLC '94, pages 144–149, Stroudsburg, PA, USA, 1994. Association for Computational Linguistics.

[29] Kemal Oflazer, Bilge Say, Dilek Zeynep Hakkani-Tür, and Gökhan Tür. Building A Turkish Treebank, 2003.

[30] J. R. Quinlan and Jack Mostow. Learning logical definitions from relations. In *Machine Learning*, pages 239–266, 1990.

[31] Stephen Soderland. Building a machine learning based text understanding system. In *In Proc. IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, pages 64–70, 2001.

[32] John F. Sowa. Conceptual graphs for a data base interface. *IBM Journal of Research and Development*, pages 336–357, 1976.

[33] John F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[34] Ergin Soysal. *Ontology Based Information Extraction on Free Text Radiological Reports Using Natural Language Processing Approach*. PhD thesis, METU, 2010.

[35] Ergin Soysal, Ilyas Cicekli, and Nazife Baykal. Design and evaluation of an ontology based information extraction system for radiological reports. *Comput. Biol. Med.*, 40:900–911, November 2010.

[36] R. Sproat. *Morphology and computation*. ACL-MIT Press Series in Natural Language Processing. The MIT Press, Cambridge, MA, 1992.

[37] P. Spyns. Natural language processing in medicine: an overview. *Methods of information in medicine*, 35(4-5):285–301, December 1996.

[38] Ricky K. Taira, Stephen G. Soderland, and Rex M. Jakobovits. Automatic Structuring of Radiology Free-Text Reports. *Radiographics*, 21(1):237–245, January 2001.

[39] L Tesnière. *Elements de syntaxe structurale*. Editions Klincksieck, 1959.

[40] Jordi Turmo, Alicia Ageno, and Neus Català. Adaptive information extraction. *ACM Comput. Surv.*, 38, July 2006.

[41] Deniz Yuret and Ferhan Türe. F.: Learning Morphological Disambiguation Rules for Turkish. In *Proceedings of HLT-NAACL*, 2006.

Appendix A

SAMPLE REPORTS

Two report samples are provided in this appendix. Patient names and identifier numbers were already removed from the reports. Doctor names are removed for the context of this thesis. A.1 contains a sample thoracic report, and A.2 contains a sample abdominal report.

The figures shown in data-driven method's explanation are also based on a sentence of the report in A.2.

A.1 Report 1

TORAKS BT

Klinik bilgi: Kolon ca, metastaz?

Teknik: İVKM sonrası 5 mm kalınlığında transvers kesitler alınmıştır.

Bulgular:

Mediastinal ana vasküler yapılar ile kalp büyüklüğü normaldir. Trakea ve ana bronşlar normaldir.

Mediastende ve her iki hiler bölgede kitle ya da lenfadenopati saptanmamıştır. Sağ üst lob lateral kesiminde 6x4.5 mm boyutlarında sağ minör fissür düzeyinde 5x3 mm boyutlarında 2 adet nodül mevcuttur.

Sağ akciğer apeksinde 5 mm çapında sol üst lobda daha küçük çapta birkaç adet nodül mevcuttur.

Kemik yapılarda destrüksiyon yoktur.

Sonuç: Bilateral üst loblarda milimetrik nodüller (benign-metastaz ayırımı yapılamamaktadır).

3 ay sonra kontrol BT ile takibi önerilir.

Dr. *(Name of medical doctor)*
Prof.Dr. *(Name of medical doctor)*

Hacettepe Üniversitesi Hastaneleri Radyoloji Anabilim Dalı'nın radyolojik inceleme raporudur.

A.2 Report 2

ÜST VE ALT ABDOMEN BT

Klinik bilgi: Peritoneal karsinomatosiz, akut karın ?, primer tümör odağı?

Teknik: İntravenöz ve oral yoldan kontrast madde verilmesinden sonra, venöz fazda, abdominal transvers düzlemde 5 mm kalınlığında aralıksız kesitler alınmıştır.

Karşılaştırma; 19.09.2006 tarihli abdomen BT ile yapılmıştır.

Bulgular: Karaciğer konturlarında belirgin indentasyon oluşturan kapsüler yerleşimli yer yer kalınlığı 1,5-2 cm'ye ulaşan peritoneal implantlar izlenmektedir. Bu yapı kılıf şeklinde karaciğerin etrafını sarmaktadır. Aynı zamanda abdomen içerisinde pariyetal peritonda ve diğer bölgelerde izlenmektedir. Karaciğer içerisinde fokal lezyon saptanmamıştır. Safra kesesi izlenmemiştir (kolestektomi?). İntra ve ekstrahepatik safra kanallarında belirgin dilatasyon mevcut değildir. Benzer kitle yapısı ve sıvı birikimi omental bursada da izlenmektedir ve mide etrafını sarmıştır. Omentumda yaygın kitle oluşumu söz konusudur. Kitlenin yaygınlığı bir evvelki BT ile karşılaştırıldığında çok belirgin değişiklik göstermemektedir. Ancak minimal sıvı artışı söz konusudur.

Dalak yine benzer kitleyle çevrelenmiş olup konturları ileri derecede irregülerdir.

Pankreasın baş, gövde ve kuyruk kesimi normal olarak değerlendirilmiştir. Splenik ven, süperior mezenterik arter ve ven patenttir.

Her iki sürrenal bez normaldir.

Her iki böbrekte en büyüğü sol böbrekte 8,5 cm çaplı olmak üzere basit kistler izlenmektedir.

Sol böbrekte tanımlanan kist bir önceki BT'ye göre minimal büyüme göstermektedir.

Abdominal aortada ve ilyak arterlerde yaygın aterosklerotik değişiklikler mevcuttur. Çölyak trunkus ve SMA orijininde yaygın aterosklerotik kalsifik plaklar vardır. Aortik bifürkasyo düzeyinin hemen üzerinde yaklaşık 2,5x1,5 cm boyutlarında aorta kaval lenfadenopati yada benzer kitlenin buraya infiltrasyon ile uyumlu olabilecek dansite artışı izlenmektedir. Parailyak lenfadenopati mevcut değildir. Pelvik lenfadenopati saptanmamıştır.

Mesane doğaldır. Prostat normal boyutlardadır.

Opasifiye olmuş bağırsak segmentlerinde intraluminal patoloji saptanmamıştır.

Kesitlere dahil kemik yapılarda litik lezyon saptanmamıştır.

Mekanik obstrüksiyon bulgusu yoktur.

Sonuç: Eylül 2006 tarihli BT ile karşılaştırıldında bulguların stabil olduğu saptanmıştır, belirgin progresyon izlenmemiştir.

Dr. *(Name of medical doctor)*
Prof. Dr. *(Name of medical doctor)*

Hacettepe Üniversitesi Hastaneleri Radyoloji Anabilim Dalı'nın radyolojik inceleme raporudur.

Appendix B

EXTRACTION DATASET

A subset of the raw dataset used for extraction is provided in this appendix. Each sentence is prefixed by a report identifier and a section identifier where the sentence is seen. *Klinikbilgi* is the clinical information section, *bulgular* is the *Findings* section and *sonuclar* is the *Conclusion* section.

The sentences are tokenized.

Each sentence is followed by a set of lines, each line being a tagged conceptual relation. In cases the same word string is repeated twice in the sentence, the word may be followed by an index in the conceptual relation.

If a comma has to be used within a concept, it is escaped with a backslash.

B.1 Subset of raw data set

```
1019:bulgular:Her iki böbreğin konturları , boyutları ve lokalizasyonu doğaldır
sifat/konum,tamlayan=Her iki,tamlanan=böbreğin
tamlama,tamlayan=böbreğin,tamlanan=konturları
tamlama,tamlayan=böbreğin,tamlanan=boyutları
tamlama,tamlayan=böbreğin,tamlanan=lokalizasyonu

1019:bulgular:Her iki böbrek toplayıcı sisteminde , bilateral üreter traselerinde ve mesane de taş ile ile uyumlu
 bulgu saptanmamıştır
sifat/konum,tamlayan=Her iki,tamlanan=böbrek toplayıcı sisteminde
sifat/konum,tamlayan=bilateral,tamlanan=üreter traselerinde
yer,nerede=böbrek toplayıcı sisteminde,ne=taş
yer,nerede=bilateral üreter traselerinde,ne=taş
yer,nerede=mesane de,ne=taş
yer,nerede=mesane de,ne=taş
anlamsal,bag=taş,anlam=uyumlu bulgu

1019:bulgular:Abdominal aortada aterom plak kalsifikasyonları izlenmektedir
sifat,tamlayan=Abdominal,tamlanan=aortada
sifat,tamlayan=aterom,tamlanan=kalsifikasyonları
```

```
sifat,tamlayan=plak,tamlanan=kalsifikasyonları
yer,nerede=aortada,tamlanan=kalsifikasyonları

1042:klinikbilgi:Sol böbrek atrofisi , sağ nefrolitiyasiz
sifat/konum,tamlayan=Sol,tamlanan=böbrek atrofisi
sifat/konum,tamlayan=sağ,tamlanan=nefrolitiyasiz

1042:bulgular:Hastada ileri düzeyde rotoskolyoz izlenmektedir
sifat,tamlayan=ileri düzeyde,tamlanan=rotoskolyoz

1042:bulgular:Sağ böbrek orta ve alt pol kaliksleri içerisinde en büyüğü 6mm çaplı multipl taş izlenmektedir
sifat/konum,tamlayan=Sağ,tamlanan=böbrek
sifat/konum,tamlayan=orta,tamlanan=pol kaliksleri içerisinde
sifat/konum,tamlayan=alt,tamlanan=pol kaliksleri içerisinde
sifat,tamlayan=böbrek,tamlanan=pol kaliksleri içerisinde
yer,nerede=pol kaliksleri içerisinde,ne=taş
sifat,tamlayan=multipl,tamlanan=taş
olcum,miktar=6mm,tur=çaplı,tamlanan=taş

1042:bulgular:Sağ üreter normaldir
sifat/konum,tamlayan=Sağ,tamlanan=üreter

1042:bulgular:Sol böbrek üst ve alt pol kalikslerinde ve parankim içerisinde en büyüğü 8mm olan taşlar
 izlenmektedir
sifat/konum,tamlayan=Sol,tamlanan=böbrek
sifat/konum,tamlayan=üst,tamlanan=pol kalikslerinde
sifat/konum,tamlayan=alt,tamlanan=pol kalikslerinde
sifat,tamlayan=böbrek,tamlanan=pol kalikslerinde
olcum,miktar=8mm,tamlanan=taşlar
yer,nerede=pol kalikslerinde,ne=taşlar
yer,nerede=parankim içerisinde,ne=taşlar

1042:bulgular:Sol hidroüreteronefroz saptanmamıştır
sifat/konum,tamlayan=Sol,tamlanan=hidroüreteronefroz

1042:bulgular:Sol böbrekte geçirilmiş pyelonefritlere sekonder yer yer parankimal incelme alanları ve skar oluşumu
 gözlenmektedir
sifat/konum,tamlayan=Sol,tamlanan=böbrekte
sifat,tamlayan=geçirilmiş,tamlanan=pyelonefritlere
konum/uzanim,nereye=pyelonefritlere,iliski=sekonder,ne=incelme alanları
sifat,tamlayan=parankimal,tamlanan=incelme alanları
konum/uzanim,nereye=pyelonefritlere,iliski=sekonder,ne=skar oluşumu

1042:sonuc:Bilateral böbrekte taşlar izlenmektedir
sifat/konum,tamlayan=Bilateral,tamlanan=böbrekte
yer,nerede=böbrekte,ne=taşlar

1043:klinikbilgi:Peritoneal karsinomatosiz , akut karın ? , primer tümör odağı ?
sifat,tamlayan=Peritoneal,tamlanan=karsinomatosiz
sifat,tamlayan=akut,tamlanan=karın
sifat,tamlayan=primer,tamlanan=tümör odağı

1043:bulgular:Karaciğer konturlarında belirgin indentasyon oluşturan kapsüler yerleşimli yer yer kalınlığı 1,5-2
 cm'ye ulaşan peritoneal implantlar izlenmektedir
yer,nerede=Karaciğer konturlarında,ne=implantlar
sifat,tamlayan=belirgin,tamlanan=indentasyon oluşturan
sifat,tamlayan=indentasyon oluşturan,tamlanan=implantlar
konum,nerede=kapsüler,nasil=yerleşimli,ne=implantlar
olcum,miktar=1\,5-2 cm'ye,tur=kalınlığı,tamlanan=implantlar
sifat,tamlayan=peritoneal,tamlanan=implantlar

1043:bulgular:Bu yapı kılıf şeklinde karaciğerin etrafını sarmaktadır
```

```
1043:bulgular:Aynı zamanda abdomen içerisinde pariyetal peritonda ve diğer bölgelerde izlenmektedir

1043:bulgular:Safra kesesi izlenmemiştir ( kolestektomi? )

1043:bulgular:Intra ve ekstrahepatik safra kanallarında belirgin dilatasyon mevcut değildir
sifat,tamlayan=Intra,tamlanan=safra kanallarında
sifat,tamlayan=ekstrahepatik,tamlanan=safra kanallarında
sifat,tamlayan=belirgin,tamlanan=dilatasyon
yer,nerede=safra kanallarında,tamlanan=dilatasyon

1043:bulgular:Benzer kitle yapısı ve sıvı birikimi omental bursada da izlenmektedir ve mide etrafını sarmıştır
sifat,tamlayan=Benzer,tamlanan=kitle yapısı
sifat,tamlayan=Benzer,tamlanan=sıvı birikimi
yer,nerede=omental bursada,ne=kitle yapısı
yer,nerede=omental bursada,ne=sıvı birikimi

1043:bulgular:Omentumda yaygın kitle oluşumu söz konusudur
yer,nerede=Omentumda,ne=kitle oluşumu
sifat,tamlayan=yaygın,tamlanan=kitle oluşumu

1043:bulgular:Kitlenin yaygınlığı bir evvelki BT ile karşılaştırıldığında çok belirgin değişiklik göstermemektedir

1043:bulgular:Ancak minimal sıvı artışı söz konusudur
sifat,tamlayan=minimal,tamlanan=sıvı artışı

1043:bulgular:Dalak yine benzer kitleyle çevrelenmiş olup konturları ileri derecede irregülerdir

1043:bulgular:Her iki böbrekte en büyüğü sol böbrekte 8,5cm çaplı olmak üzere basit kistler izlenmektedir
sifat/konum,tamlayan=Her iki,tamlanan=böbrekte
sifat/konum,tamlayan=sol,tamlanan=böbrekte[2]
yer,nerede=böbrekte,ne=kistler
yer,nerede=böbrekte[2],ne=kistler
olcum,miktar=8\,5cm,tur=çaplı,tamlanan=kistler
sifat,tamlayan=basit,tamlanan=kistler

1043:bulgular:Sol böbrekte tanımlanan kist bir önceki BT'ye göre minimal büyüme göstermektedir
sifat/konum,tamlayan=Sol,tamlanan=böbrekte
yer,nerede=böbrekte,ne=kist
sifat,tamlayan=minimal,tamlanan=büyüme

1043:bulgular:Abdominal aortada ve ilyak arterlerde yaygın aterosklerotik değişiklikler mevcuttur
sifat,tamlayan=Abdominal,tamlanan=aortada
sifat,tamlayan=ilyak,tamlanan=arterlerde
sifat,tamlayan=yaygın,tamlanan=değişiklikler
sifat,tamlayan=aterosklerotik,tamlanan=değişiklikler
yer,nerede=aortada,ne=değişiklikler
yer,nerede=arterlerde,ne=değişiklikler

1043:bulgular:Çölyak trunkus ve SMA orijininde yaygın aterosklerotik kalsifik plaklar vardır
sifat,tamlayan=yaygın,tamlanan=plaklar
sifat,tamlayan=aterosklerotik,tamlanan=plaklar
sifat,tamlayan=kalsifik,tamlanan=plaklar
yer,nerede=Çölyak trunkus,ne=plaklar
yer,nerede=SMA orijininde,ne=plaklar

1043:bulgular:Aortik bifürkasyo düzeyinin hemen üzerinde yaklaşık 2,5x1,5cm boyutlarında aorta kaval lenfadenopati
 yada benzer kitlenin buraya infiltrasyon ile uyumlu olabilecek dansite artışı izlenmektedir

1043:bulgular:Parailyak lenfadenopati mevcut değildir
sifat,tamlayan=Parailyak,tamlanan=lenfadenopati

1043:bulgular:Opasifiye olmuş bağırsak segmentlerinde intraluminal patoloji saptanmamıştır
sifat,tamlayan=Opasifiye,tamlanan=bağırsak segmentlerinde
```

```
sifat,tamlayan=intraluminal,tamlanan=patoloji
yer,tamlayan=bağırsak segmentlerinde,ne=patoloji

1043:bulgular:Mekanik obstrüksiyon bulgusu yoktur
sifat,tamlayan=Mekanik,tamlanan=obstrüksiyon bulgusu

1043:sonuc:Eylül 2006 tarihli BT ile karşılaştırıldında bulguların stabil olduğu saptanmıştır , belirgin
 progresyon izlenmemiştir
sifat,tamlayan=belirgin,tamlanan=progresyon

1044:klinikbilgi:Opere wip şant , peritonit?

1044:bulgular:Karaciğer , dalak , safra kesesi , safra yolları , pankreas , bilateral adrenal bezler ve böbrekler
 normal olarak değerlendirilmiştir
sifat/konum,tamlayan=bilateral,tamlanan=adrenal bezler

1044:bulgular:Bir önceki tetkikte mezenter kökünde ve batın içerisinde yaygın dansite artışı izlenmiştir
yer,nerede=mezenter kökünde,ne=dansite artışı
yer,nerede=batın içerisinde,ne=dansite artışı
sifat,tamlayan=yaygın,tamlanan=dansite artışı

1044:bulgular:Bu günkü BT tetkikinde mezenter kökünün tümüyle ödemli ve inflame olduğu izlenmektedir ancak
 peritondaki dansite artışı saptanmamıştır

1044:bulgular:Ayrıca mezenterik pannikülit ile uyumlu olacak şekilde mezenter kökünde en büyüğü 10mm çapında
 multipl lenf nodları izlenmiştir
yer,nerede=mezenter kökünde,ne=lenf nodları
sifat,tamlayan=multipl,tamlanan=lenf nodları
olcum,miktar=10mm,tur=çapında,tamlanan=lenf nodları
sifat,tamlayan=mezenterik,tamlanan=pannikülit

1044:bulgular:Ayrıca sol paraaortik alanda da en büyüğü 2.5cm çapında reaktif birkaç adet lenf nodu vardır
sifat/konum,tamlayan=sol,tamlanan=alanda
sifat,tamlayan=paraaortik,tamlanan=alanda
olcum,miktar=2.5cm,tur=çapında,tamlanan=lenf nodu
sifat,tamlayan=birkaç adet,tamlanan=lenf nodu
sifat,tamlayan=reaktif,tamlanan=lenf nodu
yer,nerede=alanda,ne=lenf nodu

1044:bulgular:Barsak duvarlarında ödem veya duvar kalınlaşması saptanmamıştır
yer,nerede=Barsak duvarlarında,ne=ödem
yer,nerede=Barsak duvarlarında,ne=duvar kalınlaşması

1044:sonuc:Sklerozan mezenterik pannikulit ile uyumlu bulgular
sifat,tamlayan=mezenterik,tamlanan=pannikulit
sifat,tamlayan=Sklerozan,tamlanan=pannikulit

1045:klinikbilgi:Karın ağrısı , koledokolitiazis?

1045:bulgular:Safra kesesinde taş ile uyumlu olabilecek dolum defekti saptanmamıştır
yer,nerede=Safra kesesinde,ne=taş

1045:sonuc:Normal sınırlarda üst abdomen MRG ve MRCP tetkiki

1056:bulgular:27.08.2007 tarihli BT ile karşılaştırılmıştır

1056:bulgular:Dalak normalden büyüktür , parankimi normal izlenmektedir

1056:bulgular:Paraaortik , parakaval kısa çapı 1cm'yi geçmeyen milimetrik lenf nodları izlenmektedir
sifat,tamlayan=Paraaortik,tamlanan=lenf nodları
sifat,tamlayan=parakaval,tamlanan=lenf nodları
olcum,miktar=1cm'yi,tur=çapı,tamlanan=lenf nodları
sifat,tamlayan=milimetrik,tamlanan=lenf nodları
```

```
1056:bulgular:Önceki tetkikte de bu lenf nodları görülmektedir

1056:bulgular:Bilateral inguinal bölgelerde de milimetrik lenf nodları izlenmiştir
sifat/konum,tamlayan=Bilateral,tamlanan=bölgelerde
sifat,tamlayan=inguinal,tamlanan=bölgelerde
sifat,tamlayan=milimetrik,tamlanan=lenf nodları
yer,nerede=bölgelerde,ne=lenf nodları

1056:sonuc:Splenomegali

1077:klinikbilgi:Maltoma , gaz-gaita çıkaramama , karın şişliği

1077:bulgular:Karaciğerde , büyüğü segment 5'de yaklaşık 3cm çaplı 2 adet basit kist izlenmiştir
yer,nerede=Karaciğerde,ne=kist
yer,nerede=segment 5'de,ne=kist
sifat,tamlayan=2 adet,tamlanan=kist
sifat,tamlayan=basit,tamlanan=kist
olcum,miktar=3cm,tur=çaplı,tamlanan=kist

1077:bulgular:Safra kesesi hidropik görünümdedir
sifat,tamlayan=hidropik,tamlanan=Safra kesesi

1077:bulgular:Mide ve dalak izlenmemiştir ( opere )

1077:bulgular:Her iki böbrekte büyüğü sol böbrek inferiordaki 2,5cm çaplı basit kistler dışında her iki böbreğin
 konturu , büyüklüğü , parankimi ve toplayıcı sistemi normaldir , hidronefroz izlenmemiştir
sifat/konum,tamlayan=Her iki,tamlanan=böbrekte
sifat/konum,tamlayan=sol,tamlanan=böbrek inferiordaki
yer,nerede=böbrekte,ne=kistler
yer,nerede=böbrek inferiordaki,ne=kistler
sifat,tamlayan=basit,tamlanan=kistler
olcum,miktar=2\,5cm,tur=çaplı,tamlanan=kistler

1077:bulgular:Perihepatik , intraabdominal yaygın sıvı izlenmiştir
sifat,tamlayan=Perihepatik,tamlanan=sıvı
sifat,tamlayan=intraabdominal,tamlanan=sıvı
sifat,tamlayan=yaygın,tamlanan=sıvı

1077:bulgular:İntestinal obstrüksiyon izlenmektedir
sifat,tamlayan=İntestinal,tamlanan=obstrüksiyon

1077:bulgular:Obstrüksiyon düzeyi ileum orta düzeyde olup , görünüm internal herni açısından şüphe uyandırmaktadır

1077:bulgular:Uzun ileal segmentte difüz duvar kalınlaşması olup , görünüm internal herniye sekonder iskemi
 açısından anlamlı olabilir
sifat,tamlayan=Uzun,tamlanan=ileal segmentte
sifat,tamlayan=difüz,tamlanan=duvar kalınlaşması
yer,nerede=ileal segmentte,ne=duvar kalınlaşması

1077:bulgular:İnce barsak segmentlerinin bir kısmı toraks içerisine hernie görünümdedir

1077:bulgular:Sol akciğer alt lobu tamamen atelektatik görünümdedir
sifat/konum,tamlayan=Sol,tamlanan=akciğer
sifat/konum,tamlayan=alt,tamlanan=lobu
sifat,tamlayan=tamamen,tamlanan=atelektatik

1077:bulgular:Solda 10cm'ye varan sağda minimal olmak üzere bilateral plevral efüzyon izlenmiştir
yer,nerede=Solda,ne=efüzyon
yer,nerede=sağda,ne=efüzyon
sifat,tamlayan=plevral,tamlanan=efüzyon
sifat/konum,tamlayan=bilateral,tamlanan=efüzyon
sifat,tamlayan=minimal,tamlanan=efüzyon
```

```
olcum,miktar=10cm’ye,tamlanan=efüzyon

1077:sonuc:İntestinal obstrüksiyon , internal herni açısından şüphe çeken görünüm , intraabdominal serbest sıvı ,
 uzun ileal segmentte iskemi açısından şüpheli görünüm , barsak segmentlerinin toraks içine herniasyonu , sol
 akciğerde atelektazi , solda masif olmak üzere bilateral plevral efüzyon
sifat,tamlayan=İntestinal,tamlanan=obstrüksiyon
sifat,tamlayan=internal,tamlanan=herni
sifat,tamlayan=intraabdominal,tamlanan=sıvı
sifat,tamlayan=serbest,tamlanan=sıvı
sifat,tamlayan=uzun,tamlanan=ileal segmentte
sifat/konum,tamlayan=sol,tamlanan=akciğerde
yer,nerede=akciğerde,ne=atelektazi
yer,nerede=solda,ne=efüzyon
sifat,tamlayan=plevral,tamlanan=efüzyon
sifat/konum,tamlayan=bilateral,tamlanan=efüzyon
sifat,tamlayan=masif,tamlanan=efüzyon
```