TEXT ANALYSING AND INTERROGATING

Çiğdem Yüksel, Yıldız Technical University, Department of Computer Engineering, Istanbul, cigdem_y79@hotmail.com
Hüseyin Çabuk, Yıldız Technical University, Department of Computer Engineering, Istanbul, husmanc@yahoo.com
Zeki Mocan, Yıldız Technical University, Department of Computer Engineering, Istanbul, zekmcn@hotmail.com
M.Fatih Amasyalı, Yıldız Technical University, Department of Computer Engineering, Istanbul, mfatih@ce.yildiz.edu.tr
Banu Diri, Yıldız Technical University, Department of Computer Engineering, Istanbul, banu@ce.yildiz.edu.tr

Abstract

This study has been developed in the research area of Artificial Intelligence (AI), Natural Language Processing (NLP). The goal of the study is to make some questions about a text given by the user. In the first part of the study, the text in question is parsed into its elements and translated to Prolog predicates. In the second part of the study, some questions are represented to user by analysing the Prolog predicates, and answers received from the user are examined.

Keywords

Artificial Intelligence, Natural Language Processing, Prolog, Turkish Language

1. INTRODUCTION

Since the earliest years of development in the computer technologies area, people have wanted to communicate with computers as they do between themselves. Artificial Intelligence theory has been used as a source for explaining the working of human brain. However, the latest situation doesn't seem satisfactory and sufficient.

The goal of NLP, that is a research area of AI, is to design and build a computer system that will analyse, understand, and generate natural languages. One of the easiest tasks for a NLP system is to parse a sentence to determine its syntax. A more difficult task is to determining the semantic meaning of a sentence. One of the most difficult tasks is the analysis of the context to determine the true meaning and comparing that with other text.

Turkish is a very ancient language, with a flawless phonetic, morphological and syntactic structure, and at the same time possesses a wealth of vocabulary. The fundamental features which distinguish the Ural-Altaic languages from Indo-European are as follows:

1. Vowel harmony, a feature of all Ural-Altaic tongues.
2. The absence of gender.
3. Agglutination.
4. Adjectives precede nouns.
5. Verbs come at the end of the sentence.

The first step of the study is creating web-based modular NLP tools which can run on the Internet, and NLP documentaries. These NLP tools are ***wordparser, sentenceparser, elementparser, prolog_format_converter, question_producer, answer_examiner***, respectively[1].

In the ***wordparser*** module, all words in the text are parsed according to the format of root-stem-construction affix-conjugation affix. In the ***sentenceparser*** module, complex sentences, which includes more than one verb, are separated into simple sentences and these simple sentences are related to each other in respect of connection between them. In the ***elementparser*** module, all components of sentences are defined.

In the second step of the study, by analysing sentences in input text, questions are represented to user and answers are examined. Once input text entered, all words are parsed into roots and stems. Then complex sentences are parsed into simple sentences and the relationships between them are defined. Finally, simple sentences are broken into their components and then they are translated to prolog clauses and added to prolog database. Similarly, user's answers are translated to prolog forms, then examined for validity. In the case of detecting the wrong answers, the right ones displayed to the user.

Turkish is an agglutinative language with word structures formed by productive affixations of derivational and inflectional suffixes to root words. This extensive use of suffixes causes morphological parsing of words to be rather complicated[2]. For example:
1. Evi çok güzel. (His/Her house is very beautiful.)
2. Evi gördün mü? (Have you seen the house?)

(When we analyse the function of the word 'evi' in the sentences)
In the first sentence, the affix '-i' is third singular possessive suffix (3SG-POSS) in the word 'evi'. In the second sentence, the affix '-i' is the condition suffix (CS) in the word 'evi'. Therefore, for the word 'evi' at the result of the analyse module, we get two solutions: "noun-3SG-POSS" and "noun-CS"; for the real solution the function of the word 'evi' in the sentence should be examined. In the second sentence, because of the predicate 'gördün mü?', it is understood that subject of the sentence is second singular possessive suffix. Otherwise, the word 'evi' should be subject, so the true analysis is "noun-CS". As a preliminary condition, we admit that inverted sentences mustn't be used for the accurate working of the system.

2. DEVELOPED NLP TOOLS

NLP tools, which are implemented, are ***wordparser, sentenceparser, elementparser, prolog_format_converter, question_producer, answer_examiner***, respectively.

2.1 WordParser

In the ***wordparser*** module, roots and affixes are fixed for every word in the input text. To find the possible roots in a word, the candidate root word is searched in the dictionary database[3], starting from the letter in the left most position. After each search, one letter is added to the candidate root. If it is found in the database, verification of the root is made. The rule of this verification is that the remaining part of the word must be constituted with conjugation suffixes. If verification succeeds, this root and suffixes are stored. At the end of the process, if there are more than one valid root-conjugation suffix formats, elimination steps take place. This elimination is done by examining them one more time in respect of where they exist in the sentence. After elimination, if more than one outcome remains, they are shown to the user and his/her choice is accepted.

2.2 SentenceParser

In the system, sentences, parsed into affixes and roots in the ***wordparser*** module, are parted for element parsing process. This partition process is applied for sentences which have more than one predicate. Having the sentence separated into main sentence and additional sentences, types of the connection between these sentences are determined.

After analysing the structure of the sentence of the Turkish, “partition words_suffixes” set is prepared. If the sentence in question includes one element of the set, this sentence is to be parsed. Then the system comes across two problems. These problems are; what the relation is between sentences which are obtained by parsing process, and how this relation is to be recorded.

For the first question, affixes and location of the words in the sentence are utilized, and for the second question, one number is assigned to each sentence. Relation is recorded by assigning numbers as “number/1” and “number/2” to subsentences.

Regarding the relation type, a relation file is constituted in the system and all relation information are stored in it. For example: “12/1” and “12/2” are related to each other with indirect complement.

In addition, if the first word of the sentence is the member of “partition words_suffixes” set, it is deleted and the relation type between the sentence in the question and the previous sentence is recorded depending on the first word.

For example: Ali okula gitmedi. Çünkü bugün hastaydı. → input sentences
Ali okula gitmedi. 45
Bugün hastaydı. 46
45 and 46 are related by the reason.

Table 1. The Rules of Parsing a Complex Sentence

Partition words_suffixes	**Relation Type**
ve, veya, yada, pekiştirme bağ, gerekçe bağ, karşıtlık bağ	Bağlacın kendisi(conjunction)
oysa, halbuki, rağmen, karşın	Rağmen (though)
gibi, kadar	Gibi (like)
diye, için, üzere, üzre	Amaç (purpose)
mi+ "soru değilse(if it is not a question)", dolayı, ötürü, çünkü	Sebep (reason)
iken, doğru, ile, kadar	Zarf Tümleci (adverb complement)
Gore	Göre(in respect of)
Beri	Başlama zamanı(start time)
ise, ancak, de	Özne (subject)
üzere+yeterlilik	Şart (condition)
üzere, çekimli fiil sondaysa(if conjugated verb is at the end)	Ayırma(partition)
zarffiil+whatever	Zarffiil(adverbal verb)
hal eki(condition suffix)+pekiştirme bağ, hal eki+de	Hal ekine göre(depends on the condition suffix)
sıfatfiil+whatever	Sıfat (adjective)
çekimli fiil(conjugated verb)+ancak	2. 1.nin şartı(second is the condition of the first)

2.3 ElementParser

After turning the complicated sentences into simple clauses (sentences which don't include more than one predicate), these clauses are seperated to its elements which mean "öğeler" in Turkish. In order to seperate, words and their suffixes are examined. There are six main complements in Turkish language[4] and these are named like this :

a-	Predicate	(Yüklem in Turkish)
b-	Subject	(Özne in Turkish)
c-	Accusative Object	(Belirtili Nesne in Turkish)
d-	Unaccusative Object	(Belirtisiz Nesne in Turkish)
e-	Indirect complement	(Dolaylı Tümleç in Turkish)
f-	Adverb Complement	(Zarf Tümleci in Turkish)

Predicate : Inform about work, becoming and movement and connect sentence to an idea.
Subject : Perform the things that predicate informs about and informs who does or which becomes.
Object : Object is a complement that is affected directly by the work that subject does. If verb in a sentence is transitive then that sentence can include object. If you would like to find an object in a sentence, you must ask to predicate "what" or "whom" questions.
Accusative Object : Should an object gets inform situation suffix (accusative)
Unaccusative Object : Should an object is not accusative word
Indirect Complement : The complement that informs the abletive case, the being present case or the dative case in a sentence.
Adverb Complement : These are word or word groups which completes the meaning of predicate by time, case, direction, quantity, style, provision or reason.

2.3.1 Working of ElementParser

After ***wordparser***, every word in the sentences were classified with their types and some kinds of symbols were added next to every word in order to understand which kind of words they are. In ***elementparser***, these symbols are used. For example if a word has "(ayr)" then this means the word is used for informing the leaving case and this word can be an indirect complement in this sentence. If it's necessary to constitute a rule table, it is going to be as the table 2.

Table 2. The Rules of Seperating Complements in a Sentence

Property of Complement	**Type of Complement**
If the word includes **(zm)**	Subject
If the word includes **(i)**	Subject
If the word includes **(s)** and the next word is not adjective	Subject
If the word includes **(çe)**	Subject
If the word includes **(tm)*ki(ki)**	Subject
If the word includes **(ie)**	Subject
If the word includes **(i)** and the next word is not includes **(ie)**	Subject
If the word includes **ki(ki)**	Adjective (this word is the part of the complement)
If the word includes **(bln)**	Accusative Object
If the word includes **(bsiz)**	Unaccusative Object
If the word includes **(yon)** and the next word is **kadar**	Adverb Complement
If the word includes **(yon)** and the next word is **göre**	Adverb Complement
If the next word is **için**	Adverb Complement
If the next word is **karşı**	Adverb Complement
If the word includes **(yon)** and the next word is **doğru**	Adverb Complement / Indirect Complement
If the word includes **(ayr)** and the next word is **dolayı** or **ötürü**	Adverb Complement
If the word includes **(ayr)** and the next word is **beri**	Adverb Complement
If there is **(zf)** the last of the word	Adverb Complement
If there is **(ayr)** or **(bul)** or **(yon)** at the last of the word	Indirect Complement
If the word includes **(s).**	Predicate
If the word includes **time** or **person suffix** or **nonfinite verbal form**	Predicate
If there is **ve** or **veya** or **ya da** after the word	The word is the part of the complement.
If **ile** is used after the word	Adverb Complement
If the word includes **(i)** and the next word includes **(ie)**	The word is the part of the complement
If the word includes **(tm)**	The word which first includes **(ie)** is taken and combine with this word

2.4 Conversion of Simple Sentences and Relations into Prolog Formats (prolog_format_converter)

It is quite hard to process and interrogate simple sentences and relations without putting them in particular order. Therefore, they are converted into particular format. The sentence format for all sentences:
vardir(predicate, subject, accusative object, unaccusative object, indirect complement, adverb complement, id, mood, tense)

So, it becomes a simple process to find the elements of a sentence. If some of the elements are absent in the sentence, their places are left empty in the format[5]. If the number of an element in the sentence is more than one, they are stored in a list manner. The relation format for all relations: **bag(relation_type,id1,id2)**This expression means that there is a relation of "**relation_type**" between two sentences with "**id1**" and "**id2**" ids. All data, that are user input, are converted to those two Prolog sentences. Those Prolog sentences, that are obtained, are located in Prolog database during the working of the system.

2.5 Obtaining Questions from Prolog Predicates (question_producer)

When questions are obtained from sentences which are stored in Prolog forms, Prolog database is interrogated in respect of element chosen by the user. All sentences which contain element in question are converted to question forms. In the process of converting the sentences to Prolog forms, the element is taken from sentence form in Prolog database, affix definitions are cleaned in the form. Word or word groups that defined the element in the original sentence are omitted and question word is placed in place of omitted word/words.

2.5.1 Subject

Sentence: Ali okula gitti.(Ali went to school.)
*vardir(git(f(f))*di(dgz),ali(i(i)),[],[],okul(i(i))*a (yon),[],144, '',dgz)*
When question is formed for subject, prolog database is interrogated with the expression "**vardir(_,X,_,_,_,_,_,_,_)**". The expression "ali(i(i))" taken from the form is converted to "ali". 'Ali' is omitted from the original sentence, and the question word 'Kim' is placed in place of 'Ali'. The question is "Kim okula gitti?(Who went to school?)".

There is two probabilities in choosing question word for subject: 'Kim(Who)' and 'Ne(What)'. While deciding the question word which is to be used, the file 'name.txt', prepared before, is checked whether it contains subject. If it contains the question word is 'Kim', otherwise it is 'Ne'.

2.5.2 Object

Sentence: Ali kediyi sevdi.(Ali fondled the cat)
*vardir(sev(f(f))*di(dgz),ali(i(i)),kedi(i(i))*y (kh)*i(bli),[],[],[],144, '',dgz)*
When question is formed for object, prolog database is interrogated with the expression "**vardir(_,_,X,Y,_,_,_,_,_)**". At the end of this process, 'kedi(i(i))*y(kh)*i(bli)' returns as the value of X; for Y, Null value returns due to the fact that there is no unaccusative object in the sentence. The expression 'kedi(i(i))*y(kh)*i(bli)' is converted to 'kediyi'. This word is an accusative object, so the question word 'Neyi(What)' replaced with it. The question is "Ali neyi sevdi?(What did Ali fondle?)".

While deciding the question word which is to be used, the file "name.txt" is checked whether it contains the word. If the file contains, question word is 'Kim' or 'Kimi', otherwise it is 'Ne' or 'Neyi'.

2.5.3 Indirect Complement

Sentence: Ali okula gitti.(Ali went to school.)
*vardir(git(f(f))*di(dgz),ali(i(i)),[],[],okul(i(i))*a (yon),[],144, ‘’,dgz)*
When question is formed for indirect complement, prolog database is interrogated with the expression "**vardir(_,_,_,_,X,_,_,_,_)**". At the end of this process, ‘okul(i(i))*a(yon)’ returns as the value of X. This value is converted to ‘okula’. This word is omitted from the original sentence and the question word ‘Nereye(Where)’ placed instead of it.

While deciding the question word which is to be used, the file "name.txt" is checked whether it contains the word. If the file contains it, the question word is ‘Kim(Who)’. Otherwise the file "object.txt" is checked whether it contains the word. If the file contains it, the question word is ‘Ne(What)’. Otherwise, the question word is ‘Nere(Where)’.

While determining the suffix which is to be added to question word, suffix definition at the end of the word is checked. If this suffix definition is directing suffix(yon), the suffix ‘-e’ is added to question word. If this suffix definition is divergence suffix(ayr), the suffix ‘-den’ is added to question word. If this suffix definition is existing suffix(bul), the suffix ‘-de’ is added to question word. All question words of indirect complement: Kime, Kimde, Kimden; Neye, Neyde, Neyden; Nereye, Nerede, Nereden. For the sentence in the example, the question word is ‘Nereye(Where)’. The question is "Ali nereye gitti?(Where did Ali go?)".

2.5.4 Adverb Complement

Sentence: Ali sabahleyin okula gitti. (Ali went to school in the morning.)
*vardir(git(f(f)) *di(dgz),ali(i(i)),[],[],okul(i(i))*a(yon),[sabahleyin],144, ‘’,dgz)*
When question is formed for adverb complement, Prolog database is interrogated with the expression "**vardir(_,_,_,_,_,X,_,_,_)**". At the end of this process, ‘sabahleyin’ returns as the value of X. The word ‘sabahleyin’ is omitted from the original sentence and the question word placed in place of it.
There is three probabilities in choosing question word for adverb complement: ‘ne zaman (when)’ for time adverbs, ‘ne kadar(how much)’ for quantity adverbs, ‘nasıl(how)’ for case adverbs. Original adverb is searched firstly in the file which contains quantity adverbs. Unless it is found, it is searched in another file which contains time adverbs. If it is found in one of the two files, relevant question word is chosen. Otherwise, the question word is determined as "nasıl". For the sentence in the example, the question word is ‘ne zaman’. The question is "Ali ne zaman okula gitti?(When did Ali go to school?)".

2.5.5 Predicate

Sentence: Ali okula gitti.(Ali went to school.)
*vardir(git(f(f))*di(dgz),ali(i(i)),[],[],okul(i(i))*a (yon),[],144, ‘’,dgz)*

When question is formed for indirect complement, prolog database is interrogated with the expression "**vardir(_,_,_,_,_,_,_,X)**". All elements of the sentence except for subject and adverb complement are omitted. While forming the question, subject and adverb complement and the question word which is constituted from the word "ne yap(what did …..do)" are used. To constitute the entire question word, tense and personal suffixes of the result which returns as the value of X are examined. Then, tense and personal suffixes are added to the expression "ne yap". For the sentence in the example, the question word is 'ne yaptı'. The question is "Ali ne yaptı?(What did Ali do?)".

2.6 Checking the User Answers (answer_examiner)

While questions are formed, answers are stored in two ways by the system. One of these ways is storing the answer word for the one-word answers. Another way is storing the **id** of the answer sentence. When user enters the answer, firstly it is compared with one-word answer of the system. If they match, the message "Your answer is correct." is given to the user. Otherwise, the answer of the user is parsed into its elements and converted to Prolog form. The **id** of the sentence is read from answers file. The Prolog form related to answer sentence and the Prolog form of the user answer are examined in a particular manner. This manner depends on the rule that only predicates and the elements in the questions of the formats are compared. If it fails, the message "Your answer is not correct. True answer is given in the box below." is given. Then the true answer is represented in the box to the user.

3. SYSTEM WORKING EXAMPLE

Input Text:

Ali Ankara'ya otobüsle gitti. Otobüste yaşlı bir bayanla tanıştı. Otobüsten gelirken ayağını kırdı. Bu yüzden hastahaneye kaldırıldı. Hastahane ücretini ödeyemediğinden mahsur kaldı.

(Ali went to Ankara by bus. He met an old woman in the bus. He broke his foot, getting out of the bus. Therefore he was taken to the hospital. He was stuck in the hospital due to the fact that he couldn't afford the hospital cost.)

Table 3. Analysis of the input text (on Internet by ISAPI technology)

Sample Questions	Content of Prolog database
Object Questions :	is(bayan,yaşlı,489).
Otobüsten gelirken neyi kırdı ?	is(bayan,bir,489).
(what did he break getting out of the bus?)	has(tanış,bayan,489).
Neyi ödeyemediğinden mahsur kaldı ?	vardir(otobüs(i(i))*le(birl)<git(f(f))*di(dgz),ali(i(i)),[],[],
(what couln't he afford?)	[ankara(i(i))*y(kh)*a(yon)],[],488,' ',dgz).
Adverb Complement Questions :	vardir(yaşlı(sf(sf))<bir(sf(sf))<bayan(i(i))*la(birl)<tanış(i(i))*dı(ef_dgz),'
Ne zaman ayağını kırdı ?	',[],[],[otobüs(i(i))*de(bul)],[],489,'',ef_dgz).
(when did he break his foot?)	vardir(gel(f(f))*i(ys)*r(gez)*ken(ken),' ',[],[],[otobüs(i(i))*den(ayr)],[],490:1,'
Indirect Complement Questions :	',gez).
Ali nereye otobüsle gitti ?	vardir(kır(i(i))*dı(ef_dgz),' ',[ayağ(i(i))*ı(iet3)*n(kh)*ı(bli)],[],[],[],490:2,'
(where did ali go by bus?)	',ef_dgz).
Nerede yaşlı bir bayanla tanıştı ?	vardir(kaldırıl(f(f))*dı(dgz),'
(where did he meet an old woman?)	',[],[],[hastahane(i(i))*y(kh)*e(yon)],[bu(sf(sf))<yüzden(zf(zf))],491,' ',dgz).
Nereden gelirken ayağını kırdı ?	vardir(öde(f(f))*y(kh)*e(ytk)*me(ol),'
(where was he getting out of when he broke his foot?)	',[hastahane(i(i))<ücret(i(i))*i(iet3)*n(kh)*i(bli)],[],[],[],492:1,' ',' ').
Bu yüzden nereye kaldırıldı?	vardir(kal(i(i))*dı(ef_dgz),' ',[],[],[],[mahsur(sf(sf))],492:2,' ',ef_dgz).
(where was he taken to ?)	bag(zaman,490:1,490:2).
Subject Questions :	bag(zaman,X,Y):- X=490:1:Q, Y=490:2:W.
Kim Ankara'ya otobüsle gitti ?	bag(zaman,X,Y):- X=490:1, Y=490:2:W.
(who went to Ankara?)	bag(zaman,X,Y):- X=490:1:Q, Y=490:2.

Kim otobüste yaşlı bir bayanla tanıştı ?	bag(için,490:1,490:2).
(who met an old woman in the bus?)	bag(için,X,Y):- X=490:1:Q, Y=490:2:W.
Kim otobüsten gelirken ayağını kırdı ?	bag(için,X,Y):- X=490:1, Y=490:2:W.
(who broke his/her foot getting out of the bus?)	bag(için,X,Y):- X=490:1:Q, Y=490:2.
Kim bu yüzden hastahaneye kaldırıldı ?	bag(sifat,492:1,492:2).
(who was taken to the hospital?)	bag(sifat,X,Y):- X=492:1:Q, Y=492:2:W.
Kim hastahane ücretini ödeyemediğinden mahsur kaldı ?	bag(sifat,X,Y):- X=492:1, Y=492:2:W.
(who was stuck in the hospital?)	bag(sifat,X,Y):- X=492:1:Q, Y=492:2.

4. CONCLUSION

Aim of the natural language processing is; to plan and to produce computer systems which is able to analyse, understand and constitute natural human languages. We apply natural language processing system to Turkish language and we believe that this application can help to improve studies in this area.

In our project, the given text is seperated into its affixes previously. Then complex sentences which contain sub-sentences are seperated into simple sentences. After this step, connections between these sentences are identified. Finally, simple sentences which contain only one verb are seperated into their elements and converted into Prolog predicates in order to query them. And then, this new predicates are added to Prolog database. To obtain questions from sentences which are stored in Prolog forms, Prolog database is interrogated in respect of element chosen by the user. All sentences which contain element in question are converted to question forms. Then questions are represented to user. The answers that are given by user are separated into their elements in this similar way and check their accuracy. If user gives wrong answer, the correct answer is presented to user.

5. INSUFFICIENT ASPECTS OF OUR PROJECT

We are using a dictionary database and at most three possible types for each word are included in this database. If more than one type is defined for a word, type for this word is determined in respect of usage frequencies in the text. Therefore ***elementparser*** and ***question_producer*** processes conclude with fail. So, questions which are asked to user might be wrong.
The process in order to find unqualified noun may result with fail. Sometimes if a sentence has not an exact subject (having a secret subject) our program may find unqualified noun as a subject. However, if a sentence does not have a secret subject, process and program is able to result as correct. So, questions direct to user are becoming true.

Some kind of adverbs like “böylece, öylece, şöyleki, şöylece” are not able to be classified as adverbs. Since they are not identified as adverbs at the beginning, they are usually classified as indirect complements because of the affixes they have.

Abbreviations

ytk	yeterlilik kipi	(sufficiency mood)
ol	olumsuzluk eki	(negativity suffix)
ys	yardımcı ses	(auxiliary sound)
ayr	ayrılma eki	(leaving suffix)
bul	bulunma eki	(existence suffix)
tm	tamlayan eki	(determinative suffix)
ef_dgz	ek fiil görülen geçmiş zaman eki	(affix predicate seen past tense suffix)
yon	yönelme eki	(direction suffix)
kh	kaynaştırma harfi	(joining suffix)
gez	geniş zaman	(simple present tense suffix)
dgz	görülen geçmiş zaman eki	(seen past tense suffix)
iet3	3. tekil iyelik şahıs eki	(third singular possessive suffix)
bln	hal eki	(condition suffix)

References

[1] E.Tatlı, F. Amasyalı and S.İşler, Sentence Analysis and Examining, Senior Project, Yıldız Technical University, 2001.
[2] Aksan Doğan, Anlambilim, Engin Yayınevi, İstanbul, 1999.
[3] B.Diri, "Turkish Dictionary Database", Yıldız Technical University, 1999
[4] http://www.edebiyatalemi.8m.com, Turkish Grammer.
[5] G. Gazdar and C. Mellish, *Natural Language Processing in Prolog*, Addison Wesley, 1989.
[6] D.Jurafsky, Speech and Language Processing, Prentice-Hall, New Jersey, 2000.