SEMANTIC CATEGORIZATION OF TURKISH LANGUAGE ELEMENTS

by
Onur Güngör

Submitted to the Department of Computer Engineering
in partial fulfilment of the requirements
for the degree of
Bachelor of Science
in
Computer Engineering

Boğaziçi University
June 2006

ABSTRACT

Project Name : Semantic Categorization of Turkish Language Elements

Term : 2005/2006 First and Second Semesters

Keywords : semantic categorization, classification, dictionary building from on-line sources.

Summary : In this project, we present a heuristic algorithm to build a hierarchy with the links being the hypernymy relations between the nouns in a dictionary. After obtaining an electronic version of TDK dictionary and surveying the literature on this subject, we have designed and implemented the algorithm. The main rules used in the heuristic can be divided into three groups which are rules that determine the hypernyms according to noun's surface form, rules that do so according to the noun's category indicated in the dictionary and rules that do so according to the content of the definition. The translations of the top nodes of WordNet are used to obtain a tree which can present most of the words and several levels of hierarchy. There are some inadequacies of the hierarchy due to the style of the definitions in the dictionary. These are explained in the evaluation section.

TABLE OF CONTENTS

ABSTRACT ii
1. INTRODUCTION 1
2. BUILDING AN ELECTRONIC DICTIONARY OF TURKISH 4
2.1. Considering Obtaining the Dictionary from TDK 4
2.2. Building the Dictionary on Our Own 4
2.3. Extraction Script 5
2.3.1. The Structure of the Output File 5
2.3.2. The Pseudocode of the Script 7
2.3.3. Running the Script 8
2.3.4. Miscellaneous 8
2.4. The List 9
2.5. Querying the Web Interface 9
2.6. The Structure of the Database 10
2.7. Additional Tasks 10
3. THE ALGORITHM 11
3.1. Extraction of Hypernyms 11
3.1.1. Extraction Rules 12
3.1.2. Hypernym Selection Criteria (HSC) 16
3.1.3. Synonymy Relations 16
3.2. Building the Hierarchy 17
4. EVALUATION 20
4.1. Statistics about the hierarchy 20
4.2. Incorrect identification of hypernyms 20
4.3. Absence of sense disambiguation 21
4.4. Tangled Hierarchy 22
4.5. Rough Comparison with Turkish WordNet 23
4.6. The Graph Properties of the Structure 24
4.7. Possible Uses of the Hierarchy 25
5. RELATED WORK 27
6. CONCLUSION 29
BIBLIOGRAPHY 30
REFERENCES NOT CITED 31

1. INTRODUCTION

NLP systems may require a database which stores several semantic relations [1] between the words or concepts of their domain. For example, a machine translation system can disambiguate two possible senses of a word by using the selectional restrictions imposed by the lexical units. To be more precise, when finding the correct parse of a sentence, the verb may declare that its subject must be an animate object, by using this restriction information the system can select the sense which is animate by querying the database accordingly.

There are several attempts to create such a database in the literature. For example WordNet [1] is the most famous of these studies. WordNet maintains synsets of English nouns, verbs and adjectives and the semantic relations between them. Nouns are organized into a hierarchical structure with the help of hyponymy relations between the noun synsets. Adjectives are organized mainly according to synonymy relations among them and to antonymy relations between the adjective synsets. The relations applied to nouns and adjectives cannot be easily applied to verbs, so whether these relations can be applied or if they can be, to what extent they can be applied is discussed. Additionally several relations that can satisfy the variants of lexical entailment are introduced. WordNet is the result of a work lasted about five years for the initial version. In the first version, it had about 95,600 word forms (nearly half of it being collocations) organized into 70,000 synsets. As presented, the manual development of these databases requires an excessive amount of human-time. So it is reasonable to study on devising a method to automatically build such databases.

In this report, a method for automatically categorizing the nouns in a dictionary into a hierarchical structure is presented. The nodes in this hierarchy are linked with hypernymy relations. The dictionary used as a raw database of nouns and their definitions is the electronic version of the TDK[2] dictionary. This is the first work on automatically building a hierarchical structure of Turkish words in a dictionary with the links being hypernymy

[1] These relations may include hypernymy, hyponymy, synonymy, meronymy, antonymy, etc. A hypernym of a word is a more general word including the meaning of that word. e.g. “Rose is a kind of flower.” This sentence states that 'flower' is a hypernym of 'rose'. Hyponymy is the converse of hypernymy, in the previous example 'rose' is a hyponym of 'flower'. Meronymy deals with the part-whole relationships, e.g. “Wheel is a part of automobile.” Antonyms are the opposite concepts of each other. e.g. “rich vs. poor”

[2] TDK is the national language institution found to promote language research in Turkey.

relations. We additionally try to divide the words into synsets automatically. We only look for the hypernymy and synonymy relations between nouns because these relations between other lexical categories such as adjectives and verbs are less applicable as discussed in the WordNet paper [1].

In paper [2], we see a similar work using Webster's Seventh New Collegiate Dictionary. A heuristic method is used to extract the hypernyms out of the definitions. The method simply collects the words separated with commas after 'to' in the definition of verbs. For nouns, it determines the section of the definition which must contain the hypernyms by bounding with word that must be in a prenominal position and another word that must be in a postnominal position, and it is assumed that all the words within this section of the definition are hypernyms. These hypernym-noun matchings are used to build the hiearchy thereafter.

The extraction of the hypernyms out of a definition is a difficult task for most of the time. Also because the lexicographers do not write the definitions in a way which helps to extract hypernyms, the resulting hierarchies can be incomplete and erroneous. So a common format for the machine readable dictionaries was recommended in [3]. The recommendation can be further extended to the inclusion of other types of semantic relations in the definition by the lexicographer.

The papers [4] and [5] does not only build a hierarchical structure based on hypernymy relations but further include relations such as meronymy (part-of), membership, active stative verbs and adjectives.

Contrary to the belief that the research on the automatic extraction of semantic information from dictionaries will be richened and it would begin producing satisfactory results sooner or later, the research on this subject did not reach the expected quality according to the papers [6] and [7]. This is mainly due to the inconsistencies and incomplete definitions found in the dictionaries. The paper [6] presents several types of errors inspected when trying to build these hierarchies automatically, such as 'too high attachment' (the situation when a word is attached to the more general term rather than the term that is a hyponym of that more general term), 'absent hypernyms' (when the hypernym extraction algorithm cannot find the right hypernym), 'missing overlap' (when a word has two hypernyms, the tree structure is malformed), 'OR-conjoined heads' (when the word has two or more hypernyms conjoined with 'or', this may cause wrong entailments for the hyponyms of the word) and 'circularity' (when the hypernym of a word is also a hyponym

of that word, so that A is a hypernym of B and B is a hypernym of A). For refining the result and to overcome the presented problems, paper [6] recommends using information from several dictionaries and to build the hierarchy for senses rather than words.

This report is organized as follows; in the next section we explain the development of the electronic version of the TDK dictionary in detail, in Section 3 we describe the methodology and the algorithms we have employed for building the hierarchy, and in Section 4, the evaluation of the resulting hierarchy is given. Related work on the subject is discussed in Section 5. In Section 6, we list some issues that can be studied on in the future.

2. BUILDING AN ELECTRONIC DICTIONARY OF TURKISH

2.1. Trying to Obtain the Dictionary from TDK

We had decided to use the dictionary published by Türk Dil Kurumu (TDK) considering its completeness although it had some highly debated decisions on some of the word spellings, amusing words, etc. Also we thought that if we asked the TDK for permission to give us the electronic version of the dictionary they would permit us and we would have easily obtained the electronic version of the dictionary. But that was not the case, we filed a petition for the electronic version but we have not received an answer to the date. In addition to this, we had learned that such petitions do get an answer stating that the electronic version is copyrighted and cannot be transferred out of TDK. As far as we know, TDK is an organization which is founded to support research on Turkish, and to be a centre for related activities such as dictionary maintenance, so this protocol of not disclosing the dictionary clearly contradicts with the purpose of the organization.

After realizing that we would not get a positive answer from TDK, we had tried to do a search on the net to see if we could find a reasonable alternative to TDK dictionary. But the search did not come up with so good results.

2.2. Building the Dictionary on Our Own

We had known that the TDK dictionary has a web interface which one can input a word and as a result gets the definition as in the printed dictionary. We could use this feature to build our own version of TDK dictionary. But we had to have a complete and sound word list. Tunga Güngör had a word list which mainly consisted of basic root or stem words, along with the information of which affix can be attached to this word. But there were 1500 entries in this list, which was clearly inadequate for our purpose. One attempt to this problem was to try to enrich the list by adding the appropriate affixes to each word in the initial list. But assuming this problem can be solved in a way, we have started to work on a script which will query the web interface and process it. This script is explained in 2.3. How we have solved the word list problem is explained in 2.4.

2.3. Extraction Script

At this point we had decided to postpone the word list problem a bit and start to work on a script which will retrieve the page generated for a word, process it and to insert it into a specially designed database. Before explaining the workings of the script, we present the structure of the file output by the script.

The code of the script is in the appendix disk.

2.3.1. The Structure of the Output File

Using XML in the output file made the information contained in the file easily understandable both by humans and computers.

We will describe the overall structure of an output file by an example.

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon> <!-- This is the root tag -->
     <entry>
     <!-- There is an 'entry' tag for each word. There can be
     arbitrary number of this tag in this level. -->
          <name> gabardin </name>
          <!-- Name of the entry. Other than normal
          dictionary entries, there can be 'name's which have
          paranthesis in front of the name text. e.g.
          "(birini) gafil avlamak". This tag must be present
          in every 'entry' tag. -->
          <affix>undefined</affix>
          <!—This tag gives information about the change
          occurs when the word is added an affix. e.g. name =
          gözlük -> affix = -ğü . If this information is void
          in the dictionary then the value of this tag is
          'undefined'. -->
          <lex_class>isim </lex_class>
```

```
<!-- Entry's lexical category. If this information
is void in the dictionary, in place of this tag,
<lex_class/> which indicates null value will be
present. -->
<stress>undefined</stress>
<!-- Describes the intonation when pronouncing the
words, it is present usually for foreign words. e.g.
(ga:zi:). If this information is void in the
dictionary, in place of this tag, <stress/> which
indicates null value will be present. -->
<pronunciation> Fransızca gabardine</pronunciation>
<!-- Describes the pronunciation of foreign words.
If this information is void, this tag may not be
available. -->
<origin>  Fransızca</origin>
<!-- Original language of the word. If this
information is void in the dictionary, in place of
this tag, <origin/> which indicates null value will
be present. -->
<meaning>
<!-- For each meaning entry for the word present in
the dictionary, there is a 'meaning' tag. -->
   <meaning_class>undefined</meaning_class>
   <!-- Lexical category of this meaning. -->
   <meaning_text> Sık dokunmuş bir tür ince yünlü
   veya pamuklu kumaş.</meaning_text>
   <!-- Text of the meaning. -->
   <quotation>
   <!-- If there is a quotation for this meaning.
   -->
     <author>undefined</author>
     <!-- Author of the quotation -->
     <quotation_text>undefined</quotation_text>
```

```
            <!-- Quotation text -->
          </quotation>
        </meaning>
        <atasozu_deyim_bilesik>
        <!– “Atasözü deyim ve birleşik fiiller” (this tag
        is not from the ‘gabardin’ entry, it is here for
        demonstration purposes.) -->
          <adb_entry> galeyana gelmek </adb_entry>
          <!-- ‘adb_entry’ tag is present for each entry
          given in the dictionary. -->
          <adb_entry> galeyana getirmek </adb_entry>
          <adb_entry> galeyan etmek </adb_entry>
        </atasozu_deyim_bilesik>
        <birlesik_sozler>
        <!-- “Birleşik Sözler” -->
          <bs_entry>gadretmek</bs_entry>
          <bs_entry> gadrolmak</bs_entry>
          <bs_entry> gadrolunmak</bs_entry>
        </birlesik_sozler>
        <!-- ‘atasozu_deyim_bilesik’ and ‘birlesik_sozler’
        tags are only present in entries which has this
        information in the definition in the dictionary.-->
    </entry>
</lexicon>
```

2.3.2. The Pseudocode of the Script

```
main {
    do initialization work;
    while there are words to query {
        query the word on the web interface of TDK;
        attach the returned node to the ‘lexicon’ node;
    }
```

```
     if execution successfully ended {
          output an XML file as 'lexicon' node being the root
          node;
     }
}

query {
     retrieve the page;
     process the page {
          parse the retrieved page using the HTML parser
          module;
          because there are a few page styles, change the
          execution order accordingly;
          while advancing in the parse tree, when came across
          a meaningful information, create a XML node and
          attach it to the current 'entry' node.
     }
     return the current 'entry' node, this is an XML node;
}
```

2.3.3. Running the Script

The script is run with two parameters, the first one is the name of the input file which contains a word at each line, and the second one is the name of the output file. Because it is an XML file, it is convenient to select a name which ends with '.xml'.

2.3.4. Miscellaneous

(A) The script waits about five seconds between each query refrain from stressing the TDK web server.

(B) In case the URL of the TDK web interface changes the scalar variable named 'first_part' must be changed in the query_word subroutine.

(C) The script logs the words which are not in the dictionary to a file with the name of the input file plus '.ntf' and the words which cannot be retrieved because of some error to a file with the name of the input file plus '.err'.

2.4. The List

While trying to find a way to systematically enrich the word list we have, we were also exploring the net for such a list. We have found a list that mainly consisted of stems or roots in the data files of Zemberek. Zemberek is a software project which tries to be a powerful library and applications collection for the use of Turkish NLP applications. But this list too required further processing.

While working on Zemberek to learn to use it for our purposes, we by chance realized that when you input a single letter into the query of dictionary of spelling web interface, one can get all the words that begins with that letter page by page. These words coincided with the words in the dictionary to the most available degree. So we have extracted the word lists of each letter by this way. Soon we had obtained a complete list of words in the TDK dictionary for Turkish. The code of the script which we have used for this is in the appendix disk.

2.5. Querying the Web Interface

For querying the database, we have used a batch script, which automates the running of the extraction script. The batch script runs the extraction script with each letter's word list obtained in the 2.4 as input file, and generates an appropriately named output file. These output files are kept in a different directory for each letter. The structure of the resulting database is described in 2.6. The code of the batch script is in the appendix disk.

During the run of the script, we have observed several irregularities in the TDK dictionary. But these errors were very rare, so probably they are originating from erroneous input in the TDK side. We have revised our extraction script to inspect and ignore these errors. The version included in this report is the most up-to-date version.

After the initial creation of the dictionary database files, we have extracted the 'Atasözleri, Deyimler ve Birleşik Fiiller' and 'Birleşik Sözler' parts and constructed lists from them. We have created additional XML files by using these lists as the input word list to the extraction script. Upon completing this, we had all the information in dictionary in the XML files. Finally, we have added the missing words which resulted from technical errors in the initial creation. Although there are still words not found in the TDK dictionary,

most of them are proper words, others are words that are not appropriate for a dictionary to include, or errors on the TDK side.

2.6. The Structure of the Database

Our dictionary can be thought as a collection of directories which include the related letter's definitions and additionally the extra information mentioned in 2.5. There is a separate directory for each letter. For characters that we can have problems with encodings of different file systems, we have instead used another scheme. e.g. ç -> Cx . Names of the other directories are the letters associated with them. e.g. The name of the directory which includes the files of letter 'h' is also 'h'.

Under each directory we have three files named HARF_$harf.xml, ADB_$harf.xml and BS_$harf.xml. The file HARF_$harf.xml contains the results of the query of the words we have obtained in 2.4. ADB_$harf.xml and BS_$harf.xml files are created from the lists extracted from HARF_$harf.xml as explained in 2.5. From the nature of the creation process, many entries in these files are sub entries of the entries in HARF_$harf.xml file. This situation is indicated by including a 'sub_entry' tag in the 'entry' tags of the words in these files.

The file named 'OzelIsimler.txt' contains the proper words which are not found in the TDK dictionary.

The file named 'totalnotfoundwords.txt' contains the words which are present in the lists created in 2.4 but not found in the web interface of the TDK dictionary.

2.7. Additional Tasks

After compilation of the XML files ended, we have started to write an interface to our dictionary. This work is ongoing.

3. THE ALGORITHM

To build a hierarchy with hypernymy relations as its links, we have employed two main steps. In the first step, the hypernyms of all nouns in the dictionary are collected by applying the hypernym extraction algorithm on their definitions. The extracted hypernyms are stored in an index to be used in the second step. This algorithm is outlined in Section 3.1.

In the second step, the hypernym index formed in the first step is used to build the hierarchy. Section 3.2 explains the algorithm responsible for this.

3.1. Extraction of Hypernyms

To extract the hypernyms out of a definition, we have devised a simple heuristic.

First we have divided the definition into chunks using comma character as the separator. The reason behind this division is to partition the definition for easy processing because the definitions follow a general pattern as in Table 1. In other words, the definitions usually finish with synonyms (if they have any) separated with commas, this is followed by a series of words separated with commas each ending with a possible hypernym.

Then starting from the last chunk and going to the beginning of the definition or in other words to the first chunk, several rules are applied. Application of these rules stops when a hypernym is found, this is because it was seen that many irrelevant results may be included if the algorithm did not stop at that point. It was previously said that synonyms of the word always come after the possible hypernyms of the word (when ordering is left to right), so finishing the application of the rules at this stage does not injure the extraction of synonyms.

```
name : (word* possiblehypernym,)* (synonym,)* synonym|(word*
possiblehypernym).
```

Table 1: General Pattern of Definitions

Because Turkish is an agglutinative language, it is usual not to have the hypernyms in their base forms so some of these rules first select the possible words as hypernym candidates and then applies the hypernym selection criteria (HSC) to these words. HSC is detailed in Section 3.1.2.

Name: dörtgen LexCat: isim, geometri (noun, geometry) Definition: Dört kenarlı çokgen, dörtkenar. There are two chunks here, first is 'dört kenarlı çokgen', and the second is 'dörtkenar'. The algorithm first applies HSC to 'çokgen' and because 'çokgen' is a root word 'çokgen' itself is recognized as the hypernym of the word. 'dörtkenar' is recognized as the only synonym of the word.

Table 2: Example Analysis

These extraction rules also identify the synonyms of the current word. But when a synonymy relation is found, the processing is continued with the remaining chunks unlike with hypernyms.

3.1.1. Extraction Rules

After making an observation of the definitions of the nouns in the dictionary, the following rules have been written. These rules can be divided into three groups which are rules that determine the hypernyms according to noun's form, rules that do so according to the noun's category indicated in the dictionary and rules that do so according to the content of the definition. Rule 1 and Rule 11 are the only members of the first group and the second group respectively, the other rules belong to the third group.

rule 1: If the word's text ends with “bilimi” or “Bilimi” then the hypernym of this word is “bilim”. e.g. Name: kanser bilimi LexCat: isim, tıp (noun, medicine) Definition: Kanser hastalıklarını inceleyen tıp dalı, kanseroloji. (the discipline of medicine on cancer diseases) Without Rule 1, the algorithm first identifies the word 'dalı' as the possible hypernym, and after applying HSC, the result 'dal' (discipline) is accepted as the only hypernym. But if we also apply Rule 1, 'bilim' (science) is recognized as the second hypernym of the noun phrase. The synonym of the noun phrase is 'kanseroloji'.

Table 3: Extraction Rule 1

rule 2: If the chunk ends with one of the text enclosed in double quotes below, then according to the group it belongs several different ways are taken to extract the hypernym: Group 1: “olanlardan her biri” “olanlardan biri” “olanlardan bazısı” Group 2: “her biri” “biri” “bazısı” Group 3: “W_1 CONJ W_2 her biri” “W_1 CONJ W_2 biri” “W_1 CONJ W_2 bazısı” (W_1 and W_2 being single words, CONJ “ve” or “veya”). For Group 1, the word before the matched text is processed according to the HSC. For Group 2, the algorithm accepts the hypernym as the word produced after removing the ABLATIVE MARKER and PLURAL suffixes respectively from the end of the word before the matched text. (“-lerden” suffix) e.g. Name: alet LexCat: isim (noun) Definition: Bir makineyi oluşturan ve işlemesine yardım eden **parçalardan** her biri. The ABLATIVE and PLURAL suffixes are stripped from the word in bold face respectively. The result is 'parça' (part). It is accepted as the hypernym. For Group 3, the algorithm applies HSC to W_1 and W_2 to determine the hypernym.

Table 4: Extraction Rule 2

rule 3: If the chunk ends with "(kimse)" or "kimse", the algorithm identifies the hypernym as "kişi". e.g. Name: abacı LexCat: isim (noun) Definition: Aba yapan veya satan kimse. (a person who produces or sales 'aba' which is a kind of clothing) Here the algorithm determines the hypernym as 'kişi' (person). If we did not apply Rule 3, the hypernym would be determined as 'kimse' (literally someone), which would not be appropriate.

Table 5: Extraction Rule 3

rule 4: If the chunk ends with "işi", the algorithm identifies the hypernym as "iş". 'işi' is the word 'iş' (act, work) with a suffix used when noun phrases are formed. The rule simply strips this suffix.

Table 6: Extraction Rule 4

rule 5: If the chunk ends with "tümü", the algorithm identifies the hypernym as the word produced after removing the POSSESIVE and PLURAL markers respectively ("-lerin" suffix) from the end of the word before "tümü". Rule 5 is written to handle definitions that end with phrases like literally "all of flowers" in English. In Turkish, this form becomes "çiçeklerin tümü".

Table 7: Extraction Rule 5

rule 6: If the chunk ends with "değil", the algorithm ignores this chunk and continues processing the remaining chunks. From observation, we saw that if a chunk ends with 'değil', the remaining chunks do not contain any useful information for hypernymy or synonymy.

Table 8: Extraction Rule 6

rule 7: If the chunk ends with "hepsi" (literally all), the algorithm identifies the hypernym as "grup" (group). e.g. Name: gelin alayı LexCat: isim (noun) Definition: Gelini damat evine götürmek için gelenlerin hepsi. (people who come for taking the bride to the groom's house) As in the example, the covert concept of group in the definition is handled by this rule.

Table 9: Extraction Rule 7

rule 8: If the chunk ends with "NP_1 ve NP_2" or "NP_1 veya NP_2", NP_i being a string which consists of 1 to 3 words with spaces between, the algorithm applies HSC to the last words of NP_i. Rule 8 applies HSC only to the last words of NP_i because it has been seen that the other two possible words are usually determiners or adjectives.

Table 10: Extraction Rule 8

rule 9: If the chunk ends with "vb.", then the algorithm applies HSC to the word before "vb.". If the "vb." is found elsewhere in the chunk, then the algorithm ignores all the remaining chunks. The decision to ignore the remaining chunks stems from the observation that nearly always when "vb." is not at the end of the chunk, the remaining chunks do not include the hypernyms. e.g. Name: acı (pain) LexCat: isim (noun) Definition: Ölüm, yangın, deprem vb. olayların yarattığı üzüntü, keder, elem. In this example definition, the underlined text is ignored, only the remaining parts are processed. The ignored text usually consists of examples of the concept right after 'vb.', here 'olay' is the stripped version of 'olayların' and means incident in English. As in line with the analysis the underlined text means literally 'death, fire, earthquake'. But they are not hypernyms of the noun. The correct hypernym is 'üzüntü' (distress). 'keder' and 'elem' are synonyms.

Table 11: Extraction Rule 9

rule 10: If the chunk ends with a word other than the other rules seek, first the entire chunk is looked up in the dictionary in order to check whether it is a synonym or not, if it is not, we apply HSC to the last word of the chunk. Rule 10 is based on the assumption that if an entire chunk is found in the dictionary, then this chunk is a synonym of the current word analyzed. e.g. Name: satım LexCat: isim, ticaret Meaning Text: Satma işi, satış. For example, 'satış' is found in the dictionary so it is recognized as a synonym of 'satım' which is correct.

Table 12: Extraction Rule 10

rule 11: If the word to be examined is of category "botanik" or "zooloji", then the ABLATIVE marker ("-den" suffix) is removed from the end of the last word of the first chunk. The hypernym is identified as the word formed of concatenation of the words in the chunk other than the last word and the word left after this removal with spaces between. If the last chunk of this word ends with a string enclosed in paranthesis, then the word before the paranthesis is applied HSC. e.g. Name: abdestbozan otu LexCat: isim, botanik (noun, botany) Definition: **Gülgillerden**, siyah ve yeşil boya çıkarılan bir bitki (Poterium spinosum). Here the ABLATIVE suffix is removed from the word in bold face and because there is no other words in the first chunk, one hypernym of the noun phrase is 'gülgiller'. The underlined word is the other hypernym. In this kind of nouns or noun phrases the first hypernym corresponds to the family of the plant or the animal.

Table 13: Extraction Rule 11

3.1.2. Hypernym Selection Criteria (HSC)

The words that are possibly hypernyms are selected by the extraction rules, then they are further processed by applying the hypernym selection criteria (HSC). HSC decides whether the candidate can be a hypernym, if it can, it processes the input word to come up with the actual hypernym. This processing mainly consists of stripping several suffixes.

The decision of whether a input word is a hypernym or not mainly depends on the analysis of morphological analyzer. We used Zemberek as a morphological analyzer. Zemberek is a library and applications package intended to provide solutions to some of the computational problems of Turkish Natural Language Processing.

Analysis of Zemberek provides us with the possible parses of the input word. From observational information, we can divide the parses of words into three groups according to the operation that will be applied. These groups are presented below in Tables 14, 15 and 16.

3.1.3. Synonymy Relations

We identify synonyms with Rule 10 as explained in Table 12 in Section 3.1.1. Synonym sets are constructed by using these synonymy relations extracted from the definitions. But these synonym sets can include words with very different meanings. This

is because the definitions of words only include references to a word, not to its particular senses. For example, look at the following synonym set:

```
(tertip, düzenleme, kura, ...)
```

In Turkish, there is a synonymy relation between `'tertip'` and `'düzenleme'`, and also between `'tertip'` and `'kura'`. But the most common sense of `'düzenleme'` is not a synonym of `'kura'`. So the synset presented above is inappropriate.

```
Analysis 1:
N(rootnoun) + X[3].
e.g.
      ... fon türü.
      ... ... N('tür')+X.
      Literally: ... fund kind.

Analysis 2:
N(rootnoun) + 3SG_POSS[4].
e.g.
      ... çetenin başı.
      ... çetenin N('baş')+3SG_POSS.
      Literally: ... band leader.

Analysis 3:
N(rootnoun)
e.g.
      ... yer.
      ... N('yer').
      Literally: ... place.

Operation:
If the analysis of the input word corresponds to one of
these analyses, the hypernym is identified as the root
of the input word.
```

Table 14: Group 1

3.2. Building the Hierarchy

The hypernym-noun pairs collected in the extraction phase are stored on a hypernym index. In this index, hypernyms are found matched with their hyponyms[5]. Because a hyponym of a hypernym itself can be a hypernym of another noun, the hypernym index

3 X symbolizes the suffix used for noun phrases (“isim tamlamaları”). The noun recieves nominative case when affixed with this suffix.

4 This is the suffix for 3rd singular possession of the nouns.

5 Hyponymy is the converse relation defined by hypernymy. If A is a hypernym of B, then B is a hyponym of A.

can be thought to have links between its elements. e.g. 'bitki' (plant) is a hypernym of 'biber' (pepper). 'biber' is a hypernym of 'yeşilbiber' (green pepper). A tree can be formed if one traverses the structure by using these links while stopping to traverse deeper when it reaches a node that has been visited before.

```
Analysis 1:
V(rootverb) + (any number of affixes) + VtoNSuff.

e.g.
      ... V('eş') + ... + VtoNSuff.
      ... eşleme.

VtoNSuff changes the lexical category of the verb to
noun. The resulting word is in nominative case.

Analysis 2:
N(rootnoun) + DerivSuff[6]

e.g.
      ... N('kitap') + DerivSuff(locative('lik'))
      ... kitaplık.

Analysis 3:
N(rootnoun) + DerivSuff

e.g.
      ... N('iyi') + DerivSuff(stative('lik'))
      ... iyilik.

Analysis 4:
N(rootnoun) + DerivSuff

e.g.
      ... N('kitap') + DerivSuff('cı')
      ... kitapçı.

Operation:
If the input word is suitable to one of the analyses
above, the hypernym is identified as the whole word.
This is because the above forms are nominative nouns.
```

Table 15: Group 2

One thing to note here is that if we simply start from 'nesne' (object) and form the tree, the tree does not cover most of the important concepts, nouns and noun phrases because in writing of the TDK dictionary, lexicographers did not need to care for specifying the hypernyms of each word clearly.

So in the light of this fact, we have decided to have a starting set of root nodes. These starting nodes are selected to be the Turkish translations of the top nodes in WordNet [8].

---

6 DerivSuff symbolizes the set of all derivational suffixes. Further definition is found in the arguments of it.

The translations with no hyponyms are discarded from the set of root nodes. These nodes are the nodes selected with '*' in Table 17. This kind of usage provided us with a tree which can present most of the words and several levels of hierarchy. The paper [9] had used this idea for building Catalan and Spanish WordNets.

```
Analysis 1:
V(rootverb) + (any number of affixes) + VtoNSuff +
3SG_POSS.

e.g.
      ... V('konuş') + VtoNSuff('ma') + 3SG_POSS.
      ... konuşması.

Operation:
If the input word is matched to the above analysis, the
last suffix which is the possession marker is removed,
and the resulting word is identified as the hypernym.
```

Table 16: Group 3

The tree is formed by performing a depth-first traversal of the structure starting with the root set presented in Table 17. During this traversal we stop in the nodes that are visited before to prevent loops in the structure. It is worthy to note that because we use several beginning root nodes, the hierarchy we obtain actually is not a single tree, but a collection of trees, a forest.

4. EVALUATION

Here we want to present a subset of the hierarchy as an example. Note that not only the words but phrases with two or more words can be found in the database (cf Figure 5).

4.1. Statistics about the hierarchy

There are about thirty thousand words in the hierarchy. Out of 83364 extraction attempts, 78769 of them resulted in at least one hypernym. This means that we achieved a hypernym extraction rate of 94 per cent. 60599 distinct hyponyms were added to the index in the process. The reason for the decrease of this figure to its half in the hierarchy is that the lexicographers did not pay attention to clearly construct a noun hierarchy. So the half of the nouns are left out in the hierarchy.

There are 72 levels in the hierarchy. This high value is due to the lack of sense disambiguation and to the nature of the definitions in the dictionary. If the lexicographers had the idea of categorization in their mind, this value would be at reasonable levels.

The maximum number of hyponyms a node has is 7719. This node is the node “iş”. This is somewhat expected because a lot of nouns in the TDK dictionary are nouns formed from verbs. We catch these with Group 2 of HSC.

4.2. Incorrect identification of hypernyms

The rules we have presented in Section 2.1.1 may have incorrectly predicted the hypernym of the definition. One such example follows:

kamu güvenliği .... can ve mal güvenliği.
public security .... life and holdings security.

In this definition, the algorithm applies Rule 8, and identifies two hypernyms, one is 'güvenlik' (security) and the other is 'can' (life). But clearly 'can' (life) is not a hypernym of 'kamu güvenliği' (public security). The solution of this error would be using a more complicated grammar that can select the appropriate adjectives for nouns.

WordNet Root Node	*Corresponding Root Node in our System*
{act, action, activity}	{hareket}
{natural object}	{doğal nesne}*
{animal, fauna}	{hayvan}
{natural phenomenon}	no correspondent
{artifact}	no correspondent
{person, human being}	{kişi}, {insan}
{attribute, property}	{özellik}
{plant, flora}	{bitki}
{body, corpus}	{gövde}, {vücut}
{possession}	{sahip}
{cognition, knowledge}	{kavrama}, {anlama}, {bilgi}
{process}	{süreç}, {işlem}
{communication}	{haberleşme}*
{quantity, amount}	{miktar}
{event, happening}	{olay}, {iş}
{relation}	{ilişki}
{feeling, emotion}	{duygu}, {his}
{shape}	{şekil}
{food}	{yiyecek}, {yemek}
{state, condition}	{durum}
{group, collection}	{grup}, {topluluk}, {küme}
{substance}	{madde}
{location, place}	{yer}
{time}	{zaman}, {vakit}
{motive}	{amaç}

Table 17: Corresponding Root Nodes with WordNet Root Nodes (Turkish correspondents marked with '*' (star) do not have any hyponyms in our hiearchy due to the content of the dictionary.)

4.3. Absence of sense disambiguation

Even in cases where the lexicographers wrote in a style which clearly identifies the hypernym, the sense information of the hypernym is missing by the nature of the dictionary definitions. e.g. The indicated hypernym can have more than one sense. In the absence of sense disambiguation, the hyponyms of these senses are all bound to only one node. So in the hierarchy, the hyponyms are seen as if they have similar meanings but usually this would not be the case with their most frequent senses.

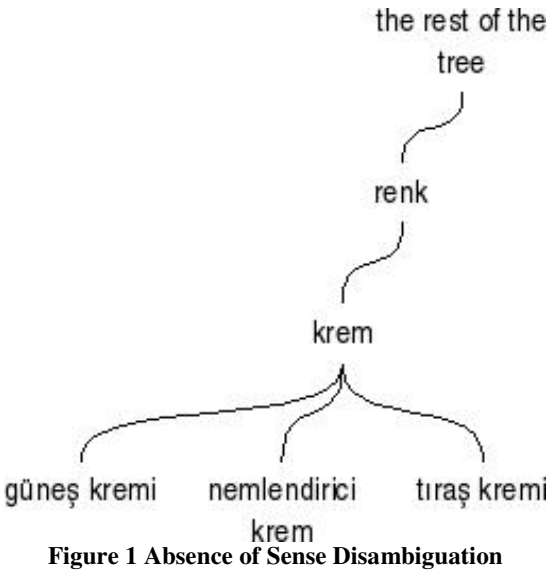

Figure 1 Absence of Sense Disambiguation

In Figure 1, 'krem' (cream) is a hyponym of 'renk' (color), and 'güneş kremi' (sun cream) is a hyponym of 'krem' (cream), but 'güneş kremi' (sun cream) is not a kind of 'renk' (color). This is because 'krem' has two different senses, one is the name of a color and the other is the substance that is used to heal various diseases, cream.

This disambiguation information is not given in the dictionary, so two different meanings of 'krem' (cream) are found in only one node under 'renk'. If the definition of 'güneş kremi' (sun cream) included the appropriate sense of 'krem' (cream), it and the other similar noun phrases would not be bound to the 'krem' (cream) node under 'renk' (color) node. They would still be bound to the 'cream' sense of 'krem' but this time that sense of 'krem' would not be bound to 'renk' (color) node.

4.4. Tangled Hierarchy

A noun or noun phrase can have more than one hypernym, this distorts the tree structure of the hierarchy by assigning two parent nodes to a node. We overcome this problem by repeating the node again in its other hypernyms. but the hyponyms of it and their hyponyms are not visited further (cf Figure 2). In the figure, 'alametifarika' has two hypernyms extracted from its definition, 'işaret' and 'özellik'. As you

can see from the figure, it is found in two places in the tree, but its children are further expanded in only one of them. This phenomenon was described before with the name "tangled hierarchy" in [10].

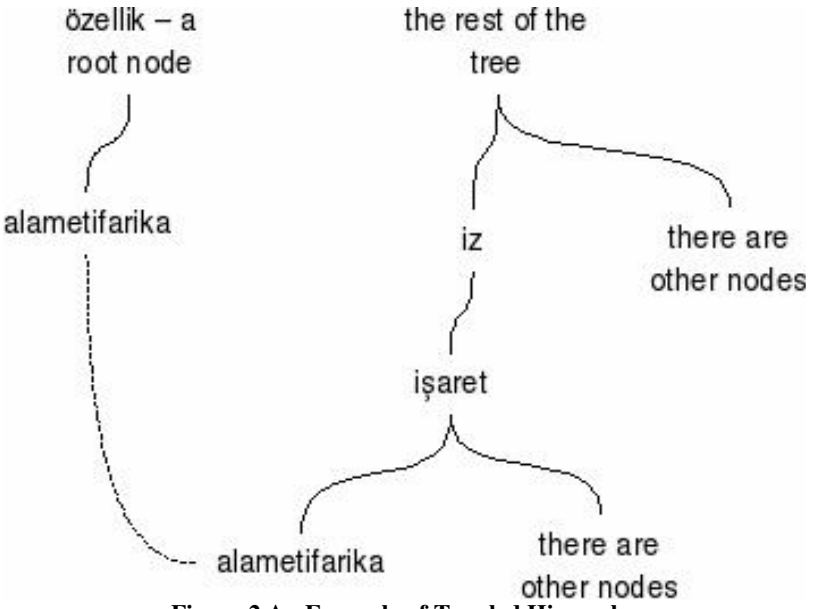

Figure 2 An Example of Tangled Hierarchy

4.5. Rough Comparison with Turkish WordNet

If we look at the partial tree from Turkish WordNet [11], with the links being hypernymy relations as in our hierarchy, we can note several points. First the structure is more concise than our hierarchy, and the nodes are sense disambiguated. But we have a broad coverage of words that are hyponyms of "`haber`" and "`bilgi`" (cf Figure 5). In contrast, the "`haber /*2`" and "`haber /2`" nodes are terminal nodes in the tree. Also the construction of our hierarchy is fully automatic which can be compared with the human intervention in the construction of Turkish WordNet.

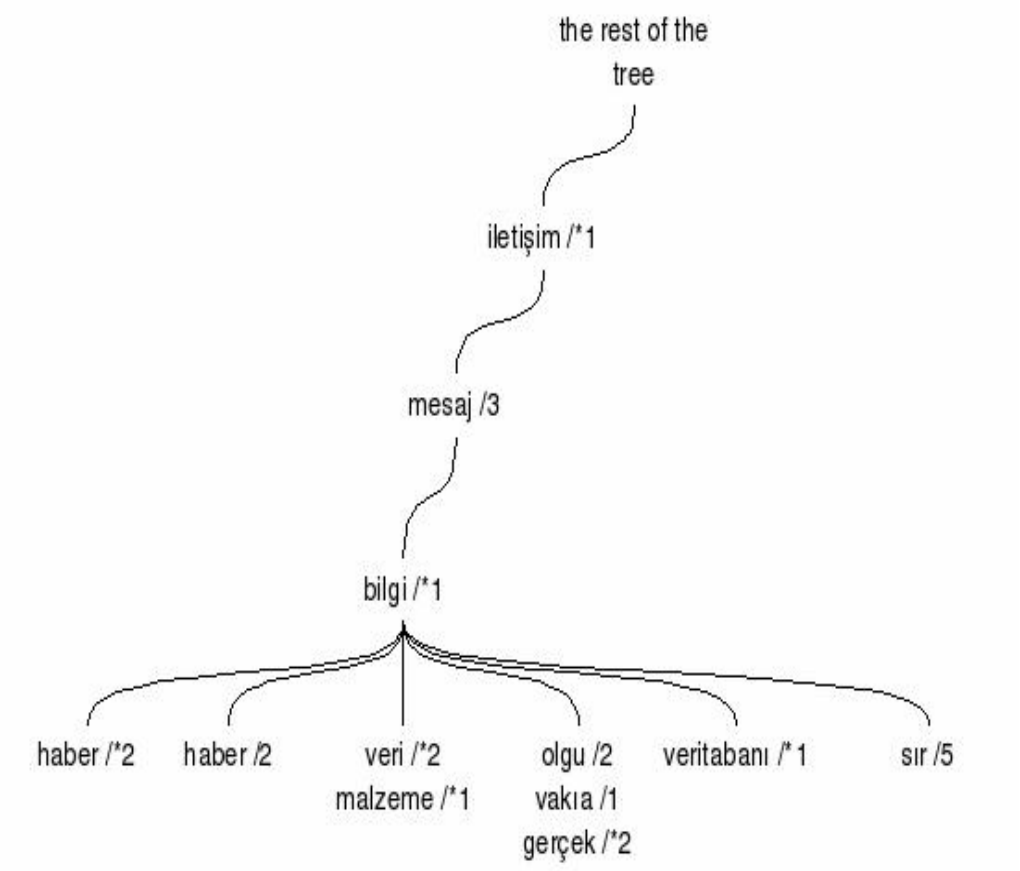

Figure 3 A Portion of Turkish WordNet

4.6. The Graph Properties of the Structure

As it can be seen on Figure 4, the hierarchy can exhibit graph properties at some points. To make sure that our structure remains a tree, we skip visiting these points and the tree under them. Therefore the link represented by the dotted line is not formed, because this would cause the dashed line between the two "kişi"'s to be realized.

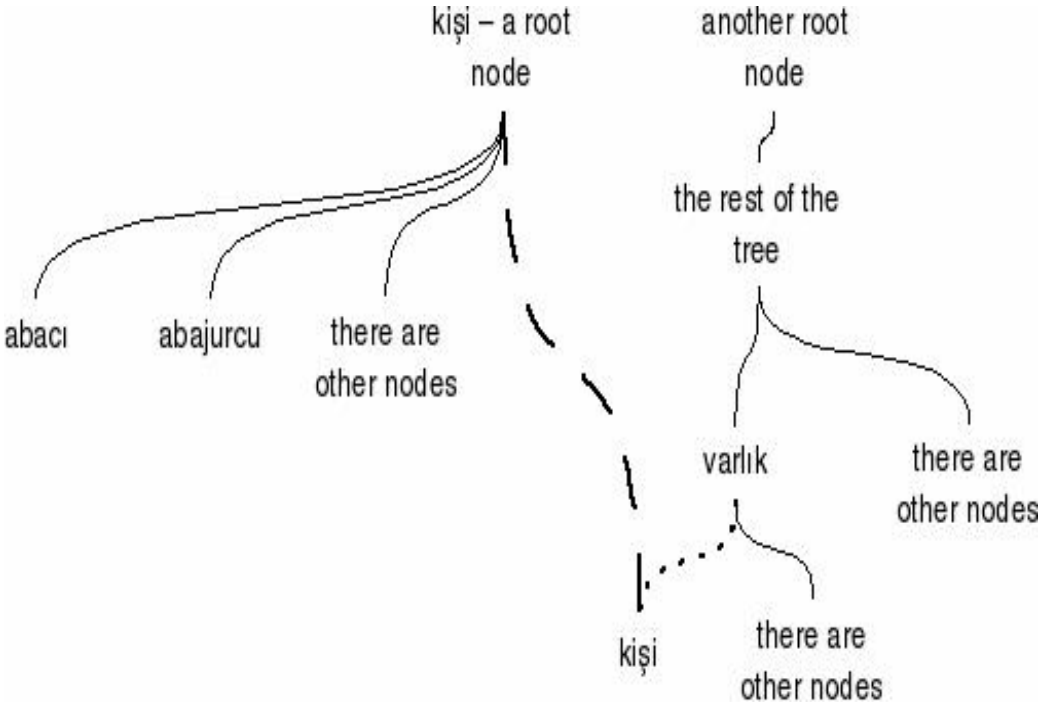

Figure 4 Graph Properties

4.7. Possible Uses of the Hierarchy

Consider the set "bülten (bulletin), duvar ilanı (wall ad), istihbarat (information, news), ilke (principle)". Suppose that we want our application to select the odd one. We can examine the relative positions of these words in the hierarchy to come to a conclusion. In the first look, we would see that "ilke" is farther from the others. So we can select it. If we need to go further, we would select "istihbarat" because "bülten" and "duvar ilanı" are siblings. (cf Figure 5)

The structure can also be used if one needs to substitute a noun or noun phrase in a text with a noun or noun phrase of similar meaning. For example, in a sentence where the object position needs a plant name or a flower name, the hierarchy can be queried to find the hyponyms of "bitki" (plant) and "çiçek" (flower) and the substitution can be made accordingly.

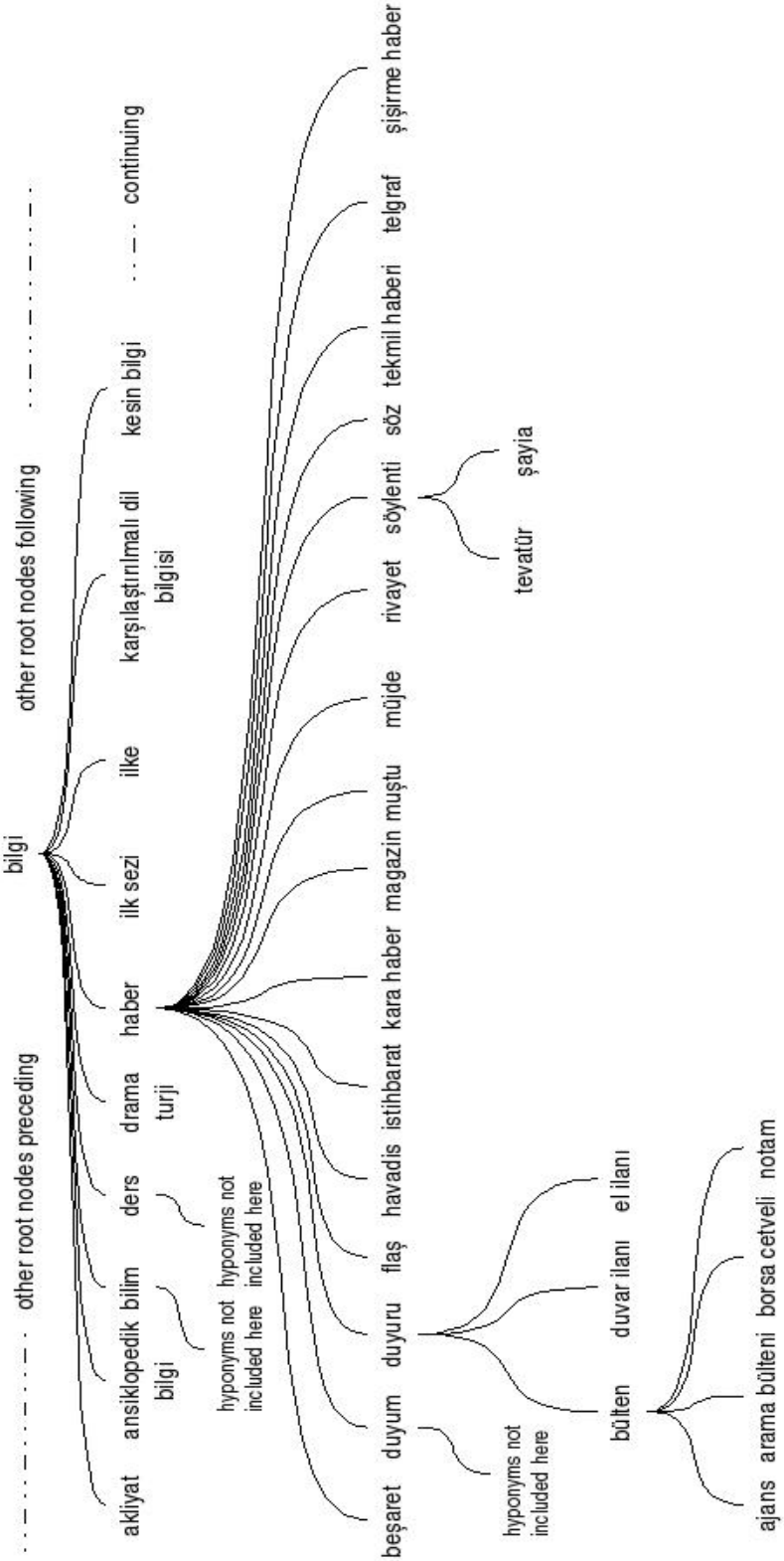

Figure 5 An Example Portion of the hierarchy

5. RELATED WORK

In this section, we will look at several papers which had built a similar structure or extracted hypernyms out of dictionary definitions in any way and will present which techniques they had used in the process.

In a work of Chodorow and Bryd [2], it is observed that the hypernym of a word is the head of the verb phrase or the noun phrase, for a verb or a noun respectively. The hypernyms of a verb are assumed to be the first word or the words when there is a conjunction after 'to' in the definition. For a noun, a region in the definition is predicted by restricting on the left and on the right by several type of words. The left edge of the region is assumed to be a word which obligatorily appears in prenominal positions. Some examples of this type of words are `a, an, the, its, two, three, ...` The right edge is similarly assumed to a word which usually occurs in postnominal positions, such as relative pronouns (e.g. that), prepositions, present participle inflected words.

In this paper, the hierarchy is built similarly by taking the transitive closure of the hypernym index created by the extraction of hypernyms from the definitions. But one difference is that the hierarchy creation process is not fully automatic. Sense disambiguation is required for each new word. If the user decides that the new word does not belong the tree being grown currently, he discards it. It may be a part of the tree in another portion of the tree.

In [6], it is concluded that the hierarchies built by processing only one dictionary will be seriously flawed an these difficulties are listed. But it is claimed that by using extraction information from different dictionaries, this flaws are to a great extent eliminated. The rationale behind this is that the reasons which make the hierarchy flawed in a dictionary probably would not be present in the other.

An analysis system which has a hierarchy of analysis patterns where less specific patterns dominate more specific ones is considered in [12]. This provides the system with more robustness because when a specific pattern could not be matched, its parents could be matched instead.

The technique in [13] uses a broad-coverage parser to obtain a parse tree of the definition and then searches for patterns that signifies several semantic relations. They use a parser of this kind because they recognize the need for several different dictionaries for a

more complete and robust semantic structure. So by using a parser, they do not need to rewrite the rules for each dictionary.

In [5], a number of specific formulae in definitions of nouns, verbs and adjectives are presented. These are simple observations such as the noun or noun coordination after “any” at the beginning of the definition of a noun is considered to be the hypernyms of that noun. This work also tries to extract semantic relations such as membership, and whether a verb is active or stative.

The paper [14] mainly discusses how to combine the semantic information from several machine readable dictionaries to build a more advanced semantic structure.

6. CONCLUSIONS

The results of our study show that a reasonable amount of semantic information can be extracted from the definitions in a dictionary. The structure have some inadequacies as presented in Section 3. This mainly originates from the lack of sense tagged hypernyms in the definitions and also from the lack of a structured style in definitions, the reason is that lexicographers do not care for these type of needs when writing the definitions.

Future work on this subject can be studying on an algorithm which can learn the heuristic by examining the definitions and making some statistical observations. Another way to go would be drawing some other heuristics for the extraction of other types of semantic relations (meronymy, antonymy, etc.) Also we could study on several different root node sets to see if we could make the hiearchy cover more nodes.

BIBLIOGRAPHY

1. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller, "Introduction to WordNet: An On-line Lexical Database," August 1993.
2. Chodorow, Martin S. and Byrd, Roy J., "Extracting semantic hierarchies from a large on-line dictionary," Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics, University of Chicago, Chicago, Illinois, 8-12 July 1985, pp.299-304, 1985.
3. Calzolari, Nicoletta, "Detecting patterns in a lexical data base," Proceedings of the 10th International Conference on Computational Linguistics, COLING'84, 2-6 July 1984, Stanford University, California, 170-173, 1984.
4. Nakamura, J., Nagao, M., "Extraction of semantic information from an ordinary English dictionary and its evaluation," COLING-88, 459-464, 1988.
5. Markowitz, Judith, Ahlswede, Thomas; and Evens, Martha, "Semantically significant patterns in dictionary definitions," Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics, New York, 112—119, 1986.
6. Ide, N., Véronis, J., "Refining taxonomies extracted from machine-readable dictionaries." In Hockey, S., Ide, N. Research in Humanities Computing 2, Oxford University Press (1993).
7. Ide, N. and Veronis, J., "Machine Readable Dictionaries: What have we learned, Where do we go?," in: Calzolari and C. Guo (eds) Proceedings of the post-coling94 international workshop on directions of lexical research, August 15-17, Beijing, 137-146, 1994.
8. Miller, George A., "Nouns in WordNet: A Lexical Inheritance System," August 1993.
9. X. Farreres, G. Rigau, and H. Rodriguez. "Using WordNet for Building WordNets," Proceedings of COLING-ACL Workshop on Usage of WordNet in Natural Language Processing Systems, Montreal, Canada, 1998.
10. Amsler, R. A., The Structure of the MerriamWebster Pocket Dictionary, Doctoral Dissertation, TR164, University of Texas, Austin, 1980.
11. Bilgin, O., Çetinoğlu, Ö., Oflazer, K., 2004, "Building a WordNet for Turkish," Romanian Journal of Information Science and Technology, Volume 7, Numbers 1-2, 2004, 163-172.
12. Alshawi, Hiyan, "Processing dictionary definitions with phrasal pattern hierarchies," American Journal of Computational Linguistics, 13(3):195-202, 1987.
13. Dolan, W., L. Vanderwende, and S. Richardson, "Automatically deriving structured knowledge bases from on-line dictionaries," Proceedings of the First Conference of the Pacific Association for Computational Linguistics, Vancouver, Canada, 5-14, 1993.
14. Antonio Sanfilippo and Victor Poznanski. "The acquisition of lexical knowledge from combined machine-readable dictionary sources." In Proceedings of the 3rd Conference on Applied Natural Language Processing (ANLP), pages 80-87, Trento, Italy, 1992.

REFERENCES NOT CITED
