A Rule-Based Morphological Disambiguator for Turkish

Turhan Daybelge
Department of Computer Engineering
Bilkent University
Bilkent 06800, Ankara, Turkey
daybelge@cs.bilkent.edu.tr

Ilyas Cicekli
Department of Computer Engineering
Bilkent University
Bilkent 06800, Ankara, Turkey
ilyas@cs.bilkent.edu.tr

Abstract

Part-of-speech (POS) tagging is the process of assigning each word of an input text into an appropriate morphological class. Automatic recognition of parts-of-speech is very important for high level NLP applications, since it would be usually infeasible to perform this task manually in practical systems. One approach to POS tagging uses morphological disambiguation which selects the most suitable morphological parse for each word from the set of parses that is assigned to that word by the morphological analyzer. Accurate POS tagging is not a simple task in general. It even becomes harder for agglutinative languages like Turkish; the number of morphological parses associated with each word in a text is usually much larger than that is for non-agglutinative languages such as English. This is due to the ambiguous nature of such languages. In this paper, we introduce an effective rule based morphological disambiguator for Turkish.

Keywords: part-of-speech tagging, morphological disambiguation.

1. Introduction

Part-of-speech (POS) tagging is the process of assigning each word of a given text into an appropriate lexical class (part of speech) such as noun, verb, adjective, etc. One approach to POS tagging is the reduction of the problem of tagging to more general morphological disambiguation problem. Once a suitable morphological parse is selected for each word from its possible morphological parses, it is trivial to detect lexical categories of words since this information is already contained in morphological parses.

In morphological disambiguation, the morphological analyzer produces all possible morphological parses for each word in the text, and a single morphological parse is tried to be selected from the set of parses assigned to that word. Unlike the ideal case, the morphological disambiguator sometimes may not select a single parse, and the selection of the best subset of parses can be aimed. Turkish part-of-speech tagger described in this paper is actually a morphological disambiguator that aims to select the best subset of the morphological parses if it cannot select a single morphological parse for a word.

Like many applications that deal with great amounts of data, it is infeasible to manually handle parts-of-speech tagging for NLP applications that require tagging of large corpora. Automatic recognition of parts-of-speech is very important for high level NLP applications such as machine translation. Although 100% accurate POS tagging is not possible in practice, highly effective systems for English are available currently. Although the effective POS taggers are available for widely studied languages such as English, the effective POS taggers are not available for the most of the languages that got less attention, and Turkish is one of these languages. In this paper, we present an effective morphological disambiguator for Turkish. The developed Turkish morphological disambiguator is planned to be used as a part of an example-based machine translation system between English and Turkish [4,5]. The developed Turkish morphological disambiguator is also integrated with a graphical user-interface so that it can be used as a morphological annotator tool for Turkish texts.

Due to the inherent morphological level ambiguity of Turkish, POS tagging and morphological disambiguation in general are much more complicated processes for Turkish. Agglutinative nature of Turkish makes the number of morphological parses for each word larger when it is compared to English. The number of possible inflectional and derivational suffixes for Turkish nouns and verbs is much higher, and this leads to the more morphological level ambiguity in Turkish words. According to [7], about 80% of Turkish words have more than one morphological parse.

There can be many different reasons for the morphological ambiguities in a Turkish word. For example, the word "*kitabın*" has the following two possible morphological parses:

kitap+Noun+A3sg+P2sg+Nom	your book
kitap+Noun+A3sg+Pnon+Gen	of the book

Here the ambiguity in the word *"kitab-ın"* is due to the phonetic similarity of the genitive suffix in the second parse and the second singular possessive suffix in the first parse. Both of them are realized as the suffix *"ın"* at surface level. Similarly, nouns with the accusative suffix and the third singular possessive suffix usually have the same surface form.

The finding the just POS tags of Turkish words will not be enough for the most NLP applications in Turkish. We have to find the actual intended morphological parse of the word. For this purpose, we have developed the Turkish morphological disambiguator presented in this paper. This morphological disambiguator tries to find the intended morphological parse of each word. If it cannot select the

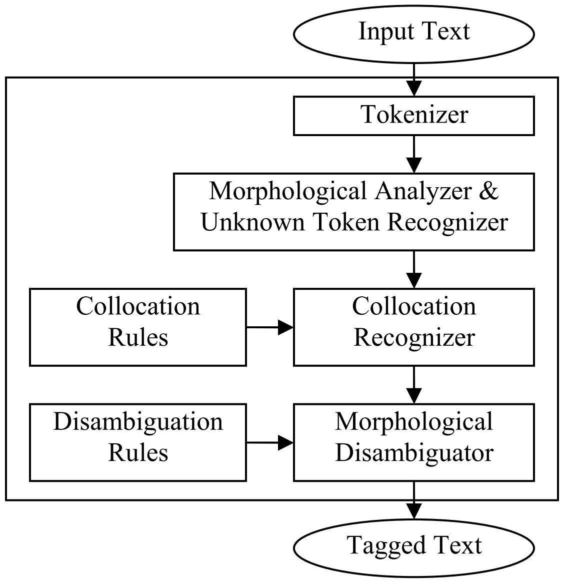

Figure 1. Architecture of the morphological disambiguator

intended morphological parse of the word, it tries to select the smallest subset of the morphological parses by eliminating some of the illegal parses.

The rest of the paper is organized as follows. Section 2 summarizes the related work in POS tagging including the previous works in Turkish POS tagging. We present the general architecture of our rule-based Turkish morphological disambiguator in Section 3. In Section 4, the performance of the presented Turkish morphological disambiguator is evaluated. We give the concluding remarks in Section 5.

2. Related Work on POS Tagging

There are two broad categories of POS tagging algorithms which are rule-based taggers and stochastic taggers. Rule based taggers contain a database of hand crafted rules that are designed to minimize ambiguity when applied in a certain order on each word in the text. Statistical POS taggers (also known as stochastic taggers), use a training corpus to calculate the likelihood of co-occurrence of all ordered pairs of tags. By training a probabilistic model such as HMM, the tagger tries to disambiguate any given new text. Since we do not have reliable morphological tagged huge corpus for Turkish, we have decided to develop a rule-based morphological disambiguator.

The earliest algorithms for automatic part-of-speech tagging were the rule-based ones. The tagger TAGGIT that was an aid in tagging the famous Brown Corpus was a rule-based one [6]. Stochastic techniques have proven to be more successful compared to pure rule-based ones. Church [3] presented a stochastic method that achieved over 95% accuracy. Also Cutting [6] presented a part-of-speech tagger based on a hidden Markov model that enables robust and accurate tagging with only a lexicon and some unlabeled training text requirements. Brill [2] presented a rule based POS tagger which used a transformation based method that learns its rules from a training corpus. Current trend in morphological disambiguation and POS tagging is blending machine learning techniques and statistic methods into rule based approaches.

Oflazer and Kuruöz [7], developed a Turkish POS tagger that uses local neighborhood constraints, heuristics and limited amount of statistical information. Oflazer and Tür [8] developed a system that combines corpus independent, linguistically motivated handcrafted constraint rules, constraint rules that are learned via unsupervised learning from a training corpus, and additional statistical information from the corpus to be morphologically disambiguated. Our morphological disambiguator is a rule-based system, and its rules are similar to the rules of the system presented in [7,8].

3. Morphological Disambiguator

Our main aim was the development of an easy to use, modern, portable and publicly available effective morphological disambiguator for Turkish. Our morphological disambiguator is purely rule-based currently, but we plan to extend it with automatic rule learning capability in the near future when the reliable morphologically tagged Turkish corpus is available. In fact, we are planning to use the developed morphological disambiguator as an annotation tool in the creation of this kind of corpus.

The developed rule-based morphological disambiguator is implemented in Java programming language, and it communicates with Turkish morphological analyzer that is developed in PCKIMO environment [1]. Our morphological disambiguator has an easy to use graphical user interface but can also be used as a command line tool. The main architecture of the morphological disambiguator is given in Figure 1. It takes an input Turkish text and produces the morphologically tagged text.

Our morphological disambiguator takes an input text, and the input text is first divided into its tokens by the tokenizer. In this way, the text is represented as a sequence of tokens. Then the morphological analyzer is run on each token and a list of morphological parses is associated with each word. Then the unknown word recognizer is run for those tokens for which the morphological analyzer has returned an empty list. The unknown token recognizer associates each unknown word with a set of morphological parses. Then collocation recognizer detects the word sequences that constitute some special meaning when they are used together, and packs them into composite tokens. Lastly the morphological disambiguator is run on the token sequence, which detects and eliminates improper morphological parses using context sensitive rules.

In our system, we have used a morphologic analyzer for Turkish that is developed using PC-KIMMO environment. The morphological level description of Turkish that was previously used in the Xerox INFL system that is developed at Bilkent University [9] has been recently ported to PC-KIMMO environment. The re-implemented system has more root words, and can handle some extra constructs such as different number constructs. The total number of the root words is more than 30,000.

After the tokenization of an input text, the tokens created by the tokenizer are sent to the morphological analyzer. After the morphological analysis, each token is assigned one or more morphological parses. For example, the results of the morphological analyzer for the token "*yarışmada*" are as follows.

1. yarış+Verb+Pos^DB+Noun+Inf2+A3sg+Pnon+Loc
2. yarış+Verb+Pos+ASPECT*PR-CONT+A3sg

After the morphological analysis there may be some tokens that are not assigned any parses such as some foreign proper nouns. These tokens are currently handled by the unknown token recognizer module. The unknown token recognizer also uses PC-KIMMO as a backend. In order to find suitable morphological parses for unknown tokens, it applies some root substitution methods that use phonetic rules of Turkish. As a simple example we can give the token "*talkshowu*" (his talkshow). The foreign word "*talkshow*" is not included in the lexicon of the morphological analyzer, so it is an unknown token. After the morphological analyzer and the unknown token recognizer, all of the tokens of the input text are associated with a set of parses.

3.1 Collocation Recognizer

The collocation recognizer takes the morphologically analyzed text and tries to detect and combine certain lexicalized and non-lexicalized collocations. The need for such a processing arises from the fact that a group of words, when appeared subsequently in a sentence, may behave as a multiword construct with a totally or partially different function compared to its individual members in that sentence. A typical example is the construct "*gelir gelmez*":

- *gelir*. (He comes.)
- *gelmez*. (He does not come.)
- *gelir gelmez* ayrıldık. (We left as soon as he comes.)

Here words "*gelir gelmez*", when used together, function in that sentence as an adverb whereas the words are inflected verbs when considered individually. There are a number of other non-lexicalized forms which are in general in the form w+x w+y, where w is the duplicated string of a root and certain suffixes, and x and y are possibly different sequences of other suffixes.

The collocation recognition is performed according to the rules given in the collocation rules file, which contains 334 rules currently. A collocation rule is sequence of token constraints and an action statement. If the sequence of token constraints matches a sequence of tokens in the text that is analyzed, the action in the action statement is applied. An action statement provides a template using which the collocation recognizer can combine the tokens in the matched sequence into a single composite token. For example, the rule that handles the collocation "*gelir gelmez*" is follows:

```
<collocationRule> <costraint> <parse>
_R+Verb+Pos+Aor+A3sg </parse> </constraint>
<costraint> <parse> _R+Verb+Neg+Aor+A3sg
</parse> </constraint> <action>
%1 %2+Adverb+When </action> </collocationRule>
```

A constraint does not always have to declare a parse to be matched, but also token readings can be matched. This kind of rules is especially used for detecting lexicalized collocations. It is also possible use regular expressions when writing token constraints. Token matching by regular expressions is case sensitive while the ordinary token matching is case insensitive.

3.2 Morphological Disambiguator

Morphological analysis of a Turkish word usually results in more than one morphological parse. This ambiguity is due to the agglutinative nature of the language. The morphological disambiguator module, using a set of context sensitive and handcrafted rules, aims to reduce the number of parses associated with each word.

Disambiguation is performed using two types of disambiguation rules, namely *choose* and *delete* rules. These rules are applied only if a word is in the specified context of the rule. By being in the context, we mean that the surrounding words match the constraints of the rule. A disambiguation rule must target a token, i.e. the token that this rule aims to disambiguate. A rule can also specify neighboring tokens, each described by an offset value, i.e. the relative position of the neighbor according to the target.

A high percentage of disambiguation rules in our system are similar to the rules in [7,8]. Our morphological disambiguator uses more capable and descriptive formatting for disambiguation rules, and the number of disambiguation rules in our system is higher when compared to that of the previous work [7,8]. Currently, the total number of the disambiguation rules is 342. 289 of them are choose-rules, and 53 of them are delete-rules.

Most of the choose rules in this file are motivated by the grammatical constraints of Turkish; so they are independent from the text category. When choose rules are applied to a certain word, if the constraints of the rule are satisfied, then the target token and its ambiguous neighbors are disambiguated at once. For the noun phrase "*çocuğun kitabı*" (the child's book), the morphological analyzer returns us the following parses:

çocuğun
1. çocuk+Noun+A3sg+Pnon+Gen *(correct parse)*
2. çocuk+Noun+A3sg+P2sg+Nom

kitabı
1. kitap+Noun+A3sg+Pnon+Acc
2. kitap+Noun+A3sg+P3sg+Nom *(correct parse)*

The tokens of this noun phrase can be disambiguated by the following choose-rule:

```
<chooseRule> <neighbour offset="-1">
<parse>A3sg+Gen</parse> </neighbour>
<target> <parse stemAllowed="false"> Noun+P3sg
</parse> </target> </chooseRule>
```

After applying the rule given above on this noun phrase, not only the word "*kitabı*" is disambiguated, but also the appropriate parse for its neighbor "*çocuğun"* is chosen.

Another set of rules, called *delete-rules*, are also used in the disambiguation process. Delete rules are mainly used to remove very rare parses of some common words. Delete rules only affect the word that is being disambiguated, and they work only in a non-ambiguous context. An example delete rule is given below:

```
<deleteRule> <target>
    <token>biz</token> <parse>Noun</parse>
</target> </deleteRule>
```

The rule above drops the very infrequent noun parse of the word "biz" in favor of the remaining pronoun parse.

The rules in the disambiguation rules file are grouped according to their function. They are also ordered according to their generality; the more a rule is stricter (specific), the higher in the file it would appear. The order of the rules is very important, because if the ordering is wrong, then the disambiguation will produce more wrong results.

4. Evaluation

In order to evaluate the performance of our morphological disambiguator, we created a test set. The test set consists of 15 randomly selected Turkish newspaper articles from online newspapers. First, the selected articles are hand tagged so that the results of the morphological disambiguator can be compared with these hand tagged articles in order to evaluate its results. Initially there were 2454 tokens in the test set. The human expert detected 77 collocations in the test set, and there were 2370 tokens (single or composite) after all collocations are hand tagged. 329 of these 2370 tokens are punctuation tokens, and 2041 of them were non-punctuation tokens. Each of 2370 tokens is correctly tagged with a single correct parse by the human expert. The human expert also selected a correct parse for the tokens that are unhandled by the morphological analyzer (unknown tokens).

Each token is assigned a set of morphological parses by the morphological disambiguator. We expect that one of these parses to be the correct one. A token is *fully disambiguated* if the disambiguator has dropped all parses except the correct one. We call the token *correctly disambiguated* if its multiple parses contain its correct parse.

We used the common precision and recall metrics in order to evaluate our morphological disambiguator. Precision measures the ratio of appropriate parses received from the morphological disambiguator to the total number of parses,

Table 5. The results after the morphological analyzer and unknown token recognizer

# of parses	1	2	3	4	5	6	7	8	9	10	11	12
# of tokens	1340	701	190	157	29	16	1	10	1	1	0	8

Table 6. The results after the collocation recognizer

# of parses	1	2	3	4	5	6	7	8	9	10	11	12
# of tokens	1304	674	172	155	28	16	1	10	1	1	0	8
# of corr. dis. toks.	1304	674	172	155	28	16	1	10	1	1	0	8

Number of Collocations	77
Total Number of Tokens	2370
Total Number of Parses	4226
Number of Corr. Disamg. Tokens	2370
Precision	**56.1%**
Recall	**100%**

Table 7. The results after applying choose-rules

# of parses	1	2	3	4	5	6	7	8	9	10	11	12
# of tokens	1820	382	70	72	7	5	1	6	1	0	0	6
# of corr. dis. toks.	1796	380	67	71	6	5	1	6	1	0	0	6

Total Number of Parses	3283
Number of Corr. Disamg. Tokens	2339
Precision	**71.2%**
Recall	**98.7%**

Table 8. The results after applying delete rules

# of parses	1	2	3	4	5	6	7	8	9	10	11	12
# of tokens	2010	271	56	22	3	7	0	1	0	0	0	0
# of corr. dis. toks.	1984	266	53	21	2	7	0	1	0	0	0	0

Total Number of Parses	2873
Number of Corr. Disamg. Tokens	2334
Precision	**81.2%**
Recall	**98.5%**

while the recall measures the ratio of correctly disambiguated tokens to the total number of tokens.

After the morphological analyzer and the unknown token recognizer steps of the disambiguator, there were 2454 tokens and there were 4383 parses for those tokens. The distribution of the tokens into the number of parses can be seen in Table 5.

Then, the collocation recognizer is executed and its results are given in Table 6. The collocation recognizer correctly found all of the 77 collocations. So, we can say that our collocation recognizer worked with 100% accuracy for this set. Although our collocation recognizer worked with 100% accuracy for this set, it can miss some collocations in a larger test set. We believe that our collocation recognizer may not be complete, but it is coverage is very high. According to the results given in Table 6, the parses of each token contain its correct parse (100% recall), and 56.1% of the all parses in the result set are correct (56.1% precision). The results in Table 6 also indicate that the average number of parses per token is 1.78 (=2370/4226), and a token can have maximum 12 parses. These measurements are the values before the disambiguation process.

We measured the precision and recall levels after applying choose and delete rules. The results after applying choose and delete rules are given in Tables 7 and 8. The precision increases from 56.1% to 71.2% by applying the choose rules by only sacrificing a small recall amount of 1.3%. The average number of parses per token also drops to 1.39 after the application of choose rules.

Finally, we apply delete rules in order to drop rare parses of tokens and achieve a precision of 81.2% and the recall becomes 98.5%. The average number of parses per token also drops to 1.21 after the application of delete rules. This is the overall performance of our morphological disambiguator. As a result, our disambiguator reduces the level of ambiguity from 1.78 parses per token to 1.21 parses per token with 81.2% precision and 98.5% recall values.

In general, precision and recall are inversely proportional to each other, i.e. it is usual to sacrifice from recall in order to improve precision. As it can be seen from the results, the decrease in recall is small when compared to the much significant increase in the precision.

5. Conclusion

In this paper, we introduced our effective rule-based morphological disambiguator for Turkish. Part-of-speech tagging is one of the low level disambiguation problems of NLP domain and although many highly accurate algorithms are available today, it still remains as an open research area especially for languages such as Turkish. Turkish, because of its agglutinative structure, has a higher ambiguity in the morphological level when compared to English. The morphological disambiguation of Turkish texts will reduce the burden in higher level NLP applications such as machine translation [4,5].

An advantage of our morphological disambiguator is that it uses a very flexible rule format for both the collocation recognition and the morphological disambiguation processes. This enables us to easily develop more rules when need arises and fine tune the behavior of the morphological disambiguator. But manually maintaining the rule files may become cumbersome as the number of rules gets large. This is due to the fact that the order of rules affects the effectiveness of the morphological disambiguator. Today, many successful algorithms are neither purely rule-based nor statistical but follow a hybrid approach that combines the best properties of the two with some machine learning approaches. These taggers can usually learn new rules by analyzing relatively small sized training corpuses and can achieve great accuracy values. Although, the morphological disambiguator developed during this project is a pure rule-based tagger with no learning capabilities, it follows a very modular approach that can easily be extended with other capabilities such as automatic rule learning in the future.

As a future work, we are planning to morphologically tag a huge Turkish corpus using our annotator tool. The researchers can use this corpus for different applications. In fact, we are planning to extend our morphological disambiguator with the statistical and automatic rule learning capabilities using this corpus.

6. References

[1] E. L. Antworth. PC-KIMMO: A Two-level Processor for Morphological Analysis. *Summer Institute of Linguistics*, Dallas, Texas, 1990.

[2] Eric Brill, A simple rule-based part of speech tagger, *Proceedings of the third conference on Applied natural language processing*, March 31-April 03, 1992, Trento, Italy

[3] Kenneth Ward Church, A stochastic parts program and noun phrase parser for unrestricted text, *Proceedings of the second conference on Applied natural language processing*, February 09-12, 1988, Austin, Texas

[4] Ilyas Cicekli, Learning Translation Templates with Type Constraints, in: Proceedings of Example-Based Machine Translation Workshop, MT Summit X, Phuket, Thailand, September 2005, pp:27-34.

[5] Ilyas Cicekli, and H. Altay Güvenir, Learning Translation Templates from Bilingual Translation Examples, in: Recent Advances in Example-Based Machine Translation, Carl, M., and Way, A. (eds), The Kluwer Academic Publishers, Boston, 2003, pp:247-278.

[6] Doug Cutting , Julian Kupiec , Jan Pedersen , Penelope Sibun, A practical part-of-speech tagger, *Proceedings of the third conference on Applied natural language processing*, March 31-April 03, 1992, Trento, Italy

[7] Kemal Oflazer, İlker Kuruöz, Tagging and morphological disambiguation of Turkish text, *Proceedings of the fourth conference on Applied natural language processing*, October 13-15, 1994, Stuttgart, Germany

[8] Kemal Oflazer, Gökhan Tür, Combining Hand-crafted Rules and Unsupervised Learning in Constraint-based Morphological Disambiguation. *Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*, May 1996, Philadelphia, PA, USA.

[9] Kemal Oflazer, Two-level Description of Turkish Morphology, Literary and Linguistic Computing, Vol. 9, No:2, 1994.