Unsupervised Learning in Constraint-based Morphological Disambiguation

Kemal Oflazer and Gökhan Tür
Department of Computer Engineering and Information Science
Bilkent University, 06533, Bilkent, Ankara, TURKEY
{ko,tur}@cs.bilkent.edu.tr

February 12, 1996

Abstract

This paper presents a constraint-based morphological disambiguation approach that uses unsupervised learning component to discover some of the constraints it uses. It is specifically applicable to languages with productive inflectional and derivational morphological processes, such as Turkish, where morphological ambiguity has a rather different nature than that found in languages like English. Our approach starts with a set of corpus-independent hand-crafted rules that reduce morphological ambiguity (hence improve precision) without sacrificing recall. It then uses an untagged training corpus in which all lexical items have been annotated with all possible morphological analyses, incrementally proposing and evaluating additional (possibly corpus dependent) constraints for disambiguation of morphological parses using the constraints imposed by unambiguous contexts. These rules *choose parses* with specified features. It then learns in an unsupervised manner, additional rules for *removing parses* with certain features. In certain respects, our approach has been motivated by Brill's recent work [3], but with the observation that his transformational approach is not directly applicable to languages like Turkish. Our results indicate that using hand-crafted rules and rules learned to choose, we can attain a recall of 99.08% and a precision of 88.08% with 1.119 parses per token, on the training text. When rules learned to delete are used in addition to these, we can attain a recall of 96.76% and a precision of 92.05% and 1.051 parses per token on the training text. On previously unseen text, we can attain a recall of 98.04% and a precision of 86.23% with 1.137 parses per token using just the hand-crafted rules and rules learned to choose. When rules learned to delete are used we can attain a recall of 96.99% and a precision of 88.13% and 1.100 parses per token.

1 Introduction

Automatic morphological disambiguation is a very crucial component in higher level analysis of natural language text corpora. Morphological disambiguation facilitates parsing, essentially by performing a certain amount of ambiguity resolution using relatively cheaper methods (e.g.,[7]). There has been a large number of studies in tagging and morphological disambiguation using various techniques. Part-of-speech tagging systems have used either a statistical approach where a large corpora has been used to train a probabilistic model which then has been used to tag new text, assigning the most likely tag for a given word in a given context (e.g., Church [4], Cutting *et al.* [5], DeRose [6]). Another approach is the rule-based or constraint-based approach, recently most prominently exemplified by the Constraint Grammar work [8, 15, 16, 17], where a large number of hand-crafted linguistic constraints are used to eliminate impossible tags or morphological parses for a given word in a given context. Brill [1, 2] has presented a transformation-based learning approach, which induces rules from tagged corpora. Recently he has extended this work so that learning can proceed in an unsupervised manner using an untagged corpus [3]. Levinger *et al.* [11] have recently reported on an approach that learns morpho-lexical probabilities from untagged corpus and have the used the resulting information in morphological disambiguation in Hebrew.

In contrast to languages like English, for which there is a very small number of possible word forms with a given root word, and a small number of tags associated with a given lexical form, languages like Turkish or Finnish with very productive agglutinative morphology where it is possible to produce thousands of forms for a given root word, pose a challenging problem for morphological disambiguation. In English, for example, a word such as *make* or *set* can be verb or a noun. In Turkish, even though there are ambiguities of such sort, the agglutinative nature of the language usually helps resolution of such ambiguities due to restrictions on morphotactics. On the other hand, this very nature introduces another kind of ambiguity, where a lexical form can be morphologically interpreted in many ways, some with totally unrelated roots and morphological features, as will be exemplified in the next section.

Our previous approach to tagging and morphological disambiguation for Turkish text had employed a constraint-based approach [13] along the general lines of similar previous work for English [8, 16, 17]. Although the results obtained there were reasonable, the fact that the constraint rules were hand crafted, posed a rather serious impediment to the generality and improvement of the system.

In this paper we present a constraint-based morphological disambiguation approach that uses unsupervised learning component to discover some of the constraints it uses. It is specifically applicable to languages with productive inflectional and derivational morphological processes, such as Turkish, where morphological ambiguity has a rather different nature than that found in languages like English. Our approach starts with a set of corpus-independent hand-crafted rules that reduce morphological ambiguity (hence improve precision) without sacrificing recall. It then uses an untagged training corpus in which all lexical items have been annotated with all possible morphological analyses, incrementally proposing and evaluating additional (possibly corpus dependent) constraints for disambiguation of morphological parses using the constraints imposed by unambiguous contexts. These rules choose parses with specified features. It then learns, in an unsupervised manner, additional rules for removing parses with certain features. In certain respects, our approach has been motivated by Brill's recent work [3], but with the observation that his transformational approach is not directly applicable to languages like Turkish, where tags associated with forms are not predictable in advance.

In the following sections, we present an overview of morphological disambiguation highlighted with examples from Turkish. We then present the details of our approach and results. We finally conclude after a discussion and evaluation of our results.

2 Tagging and Morphological Disambiguation

In almost all languages, words are usually ambiguous in their parts-of-speech or other lexical features, and may represent lexical items of different syntactic categories, or morphological structures depending on the syntactic and semantic context. Part-of-speech (POS) tagging involves assigning every word its proper part-of-speech based upon the context the word appears in. In English, for example a word such as *set* can be a verb in certain contexts (e.g., He *set* the table for dinner) and a noun in some others (e.g., We are now facing a whole *set* of problems).

In Turkish, there are ambiguities of the sort above. However, the agglutinative nature of the language usually helps resolution of such ambiguities due to the restrictions on morphotactics. On the other hand, this very nature introduces another kind of ambiguity, where a whole lexical form can be morphologically interpreted in many ways not predictable in advance. For instance, our full-scale morphological analyzer for Turkish returns the following set of parses for the word *oysa*:[1,2]

```
1. [[CAT CONN] [ROOT oysa]] (on the other hand)

2. [[CAT NOUN] [ROOT oy] [AGR 3SG] [POSS NONE] [CASE NOM]
                        [CONV VERB NONE] [TAM1 COND] [AGR 3SG]]
   (if it is a vote)

3. [[CAT PRONOUN] [ROOT o] [TYPE DEMONS] [AGR 3SG] [POSS NONE] [CASE NOM]
                        [CONV VERB NONE] [TAM1 COND][AGR 3SG]]
   (if it is)

4. [[CAT PRONOUN] [ROOT o] [TYPE PERSONAL] [AGR 3SG] [POSS NONE] [CASE NOM]
                        [CONV VERB NONE] [TAM1 COND] [AGR 3SG]]
   (if s/he is)
5. [[CAT VERB] [ROOT oy] [SENSE POS] [TAM1 DES] [AGR 3SG]]
   (wish  s/he would  carve)
```

On the other hand, the form *oya* gives rise to the following parses:

```
1. [[CAT NOUN] [ROOT oya] [AGR 3SG] [POSS NONE] [CASE NOM]] (lace)
2. [[CAT NOUN] [ROOT oy]  [AGR 3SG] [POSS NONE] [CASE DAT]] (to the vote)
3. [[CAT VERB] [ROOT oy]  [SENSE POS] [TAM1 OPT] [AGR 3SG]]  (let him carve)
```

and the form *oyun* gives rise to the following parses:

[1] Output of the morphological analyzer is edited for clarity, and English glosses have been given.

[2] Glosses are given as linear feature value sequences. The feature names are as follows: `CAT`-major category, `TYPE`-minor category, `ROOT`-main root form, `AGR` -number and person agreement, `POSS` - possessive agreement, `CASE` - surface case, `CONV` - conversion to the category following with a certain suffix indicated by the argument after that, `TAM1`-tense, aspect, mood marker 1, `SENSE`-verbal polarity, `DES`- desire mood, `IMP`-imperative mood, `OPT`- optative mood, `COND`-Conditional

```
1. [[CAT NOUN] [ROOT oyun] [AGR 3SG] [POSS NONE] [CASE NOM]] (game)
2. [[CAT NOUN] [ROOT oy]   [AGR 3SG] [POSS NONE] [CASE GEN]] (of the vote)
3. [[CAT NOUN] [ROOT oy]   [AGR 3SG] [POSS 2SG] [CASE NOM]]  (your vote)
4. [[CAT VERB] [ROOT oy]   [SENSE POS] [TAM1 IMP] [AGR 2PL]] (carve it!)
```

One can note from these and other similar examples that, the other parses that can be generated from a given form, the root words and the parses that interact, are not predictable in advance given that each noun root may give rise to thousands of possible forms and each verbal root may give rise to hundreds of thousands of forms.

On the other hand, the local syntactic context may help reduce some of the ambiguity above, as in:[3]

..	sen-in	**oy-un** ..	
	PRONOUN(you)+GEN	NOUN(vote)+POSS-2SG	
	your	*vote*	
..	**oy-un**	reng-i ..	
	NOUN(vote)+GEN	NOUN(color)+POSS-3SG	(NOUN-GEN NOUN-POSS form)
	color	*of the vote*	
..	**oyun**	reng-i ..	
	NOUN(game)	NOUN(color)+POSS-3SG	(NOUN NOUN-POSS form)
	game	*color*	

using some very basic noun phrase agreement constraints in Turkish. Obviously in other similar cases it may be possible to resolve the ambiguity completely.

There are also numerous other examples of word forms where a productive derivational process comes into play:[4]

```
    geldiGimdeki (at the time I came)

1. [[CAT VERB] [ROOT gel] [SENSE POS] (basic form)
     [CONV NOUN DIK] [AGR 3SG] [POSS 1SG] [CASE LOC] (participle form)
     [CONV ADJ REL]] (final adjectivalization by the relative (ki) suffix)
```

Here, the original root is verbal but the final part-of-speech is adjective. In general, the ambiguities of the forms that come before such a form in text can be resolved with respect to its original (or intermediate) parts-of-speech (and inflectional features), while the ambiguities of the forms that follow can be resolved based on its final part-of-speech.

Our intent is to achieve a *morphological ambiguity reduction* in the text by choosing for a given ambiguous token, a subset of its parses which are not disallowed by the syntactic context it appears

[3] With a slightly different but nevertheless common glossing convention.

[4] Upper cases in morphological output indicates one of the non-ASCII special Turkish characters: e.g., G denotes ğ.

in. It is certainly possible that a given token may have multiple correct parses, usually with the same inflectional features or with inflectional features not ruled out by the syntactic context. These can only be disambiguated usually on semantic or discourse constraint grounds. We consider a token *fully disambiguated* if it has only one morphological parse remaining after automatic disambiguation. We consider as token as correctly disambiguated, if one of the parses remaining for that token is the *correct* intended parse.

We evaluate the resulting disambiguated text by a number of metrics defined as follows [14].

$$\text{Ambiguity} = \frac{\text{Number of Parses}}{\text{Number of Tokens}}$$

$$\text{Recall} = \frac{\text{Number of Tokens Correctly Disambiguated}}{\text{Number of Tokens}}$$

$$\text{Precision} = \frac{\text{Number of Tokens Correctly Disambiguated}}{\text{Number of Parses Remaining}}$$

In the ideal case where each token is uniquely and correctly disambiguated with the correct parse, both recall and precision will be 1.0. On the other hand, a text where each token is annotated with all possible parses,[5] the recall will be 1.0 but the precision will be low. The goal is to have both recall and precision as high as possible.

3 Constraint-base Morphological Disambiguation

This section outlines our approach to constraint-based morphological disambiguation incorporating unsupervised learning component. Our system has the architecture presented in Figure 1. We take as input to the system raw Turkish text. This text is then preprocessed as explained in the next section, and then goes into the learning module. The learning module generates a sequence of rules from the given text which may then be used to disambiguate further examples of (similar) text.

3.1 The Preprocessor

The preprocessing module takes as input a Turkish text, segments it into sentences using various heuristics about punctuation, tokenizes and runs it through a wide-coverage high-performance morphological analyzer developed using two-level morphology tools by Xerox [9]. This module also performs a number of additional functions:

[5] Assuming no unknown words.

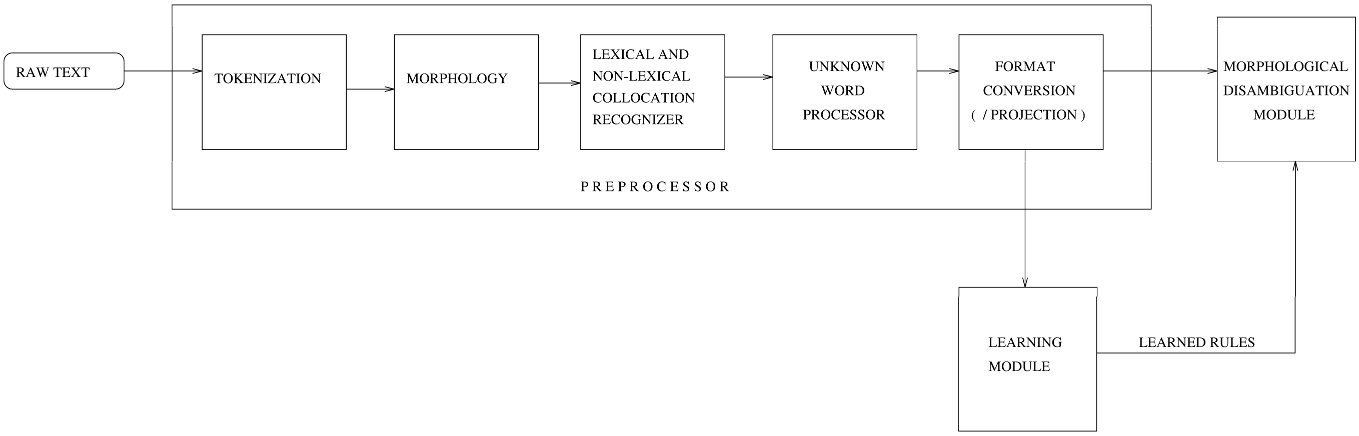

RAW TEXT
TOKENIZATION
MORPHOLOGY
LEXICAL AND NON-LEXICAL COLLOCATION RECOGNIZER
UNKNOWN WORD PROCESSOR
FORMAT CONVERSION (/ PROJECTION)
PREPROCESSOR
MORPHOLOGICAL DISAMBIGUATION MODULE
LEARNING MODULE
LEARNED RULES

Figure 1: The structure of the rule learning system.

- it groups *lexicalized collocations* such as idiomatic forms, semantically coalesced forms such as proper noun groups, certain numeric forms, etc.
- it groups any *compound verb formations* which are formed by a lexically adjacent, direct or oblique object, and a verb, which for the purposes of syntactic analysis, may be considered as single lexical item: e.g., *saygı durmak* (to pay respect), *kafayı yemek* (literally *to eat the head* – to get mentally deranged), etc.
- it groups *non-lexicalized collocations*: Turkish abounds with various non-lexicalized collocations where the sentential role of the collocation has (almost) nothing to do with the parts-of-speech of the individual forms involved. Almost all of these collocations involve duplications, and have forms like $\omega + x \ \omega + y$ where ω is the duplicated string comprising the root and certain sequence of suffixes and x and y are possibly different (or empty) sequences of other suffixes.

  The following is a list of multi-word constructs for Turkish that we handle in our preprocessor. This list is not meant to be comprehensive, and new construct specifications can easily be added. It is conceivable that such a functionality can be used in almost any language. (See [13, 10] for details of all other forms for Turkish.)

  1. duplicated optative and 3SG verbal forms functioning as manner adverb. An example is *koşa koşa*, where each lexical item has the morphological parse

     ```
     [[CAT VERB] [ROOT koS] [SENSE POS] [TAM1 OPT] [AGR3SG]]
     ```

     The preprocessor recognizes this and generates the feature sequence:

     ```
     [[CAT VERB] [ROOT koS] [SENSE POS] [TAM1 OPT] [AGR 3SG]
                [CONV ADVERB DUP1] [TYPE MANNER]]
     ```

  2. aorist verbal forms with root duplications and sense negation, functioning as temporal adverbs. For instance for the non-lexicalized collocation *yapar yapmaz*, where items have the parses

```
[[CAT VERB] [ROOT yap] [SENSE POS] [TAM1 AORIST ] [AGR 3SG]

[[CAT VERB] [ROOT yap] [SENSE NEG] [TAM1 AORIST ] [AGR 3SG]
```

respectively, the preprocessor generates the feature sequence

```
[[CAT VERB] [ROOT koS] [SENSE POS] [TAM1 AORIST] [AGR 3SG]
          [CONV ADVERB DUP-AOR] [TYPE TEMP]]
```

3. duplicated verbal and derived adverbial forms with the same verbal root acting as temporal adverbs, e.g., *gitti gideli*,
4. emphatic adjectival forms involving duplication and question clitic, e.g., *güzel mi güzel* (beautiful question-clitic beautiful–very beautiful)
5. adjective or noun duplications that act as manner adverbs, e.g., *hızlı hızlı*, *ev ev*,

This module recognizes all such forms and coalesces them into new feature structures reflecting the final structure along with any inflectional information.

- The preprocessor then converts each parse into a hierarchical feature structure so that the inflectional feature of the form with the last category conversion (if any) are at the top level. Thus in the example above for *geldiğimdeki*, the following feature structure is generated:

```
[[CAT VERB] [ROOT gel] [SENSE POS] (basic form)
     [CONV NOUN DIK] [AGR 3SG] [POSS 1SG] [CASE LOC] (participle form)
     [CONV ADJ REL]] (final adjectivalization by the relative (ki) suffix)
```

$$\Downarrow$$

$$\begin{bmatrix} \text{CAT} & \text{ADJ} \\ \text{STEM} & \begin{bmatrix} \text{CAT} & \text{NOUN} \\ \text{AGR} & \text{3SG} \\ \text{POSS} & \text{1SG} \\ \text{CASE} & \text{LOC} \\ \text{STEM} & \begin{bmatrix} \text{CAT} & \text{VERB} \\ \text{ROOT} & \text{gel} \\ \text{SENSE} & \text{POS} \end{bmatrix} \\ \text{SUFFIX} & \text{DIK} \end{bmatrix} \\ \text{SUFFIX} & \text{REL} \end{bmatrix}$$

- Finally, each such feature structure is then *projected* on a subset of its features. The features selected are
  - inflectional and certain derivational markers, and stems for open class of words,
  - roots and certain relevant features such as subcategorization requirements for closed classes of words such as connectives, postpositions, etc.

  The set of features selected for each part-of-speech category is determined by template and hence is user controllable permitting experimentation with differing levels of information. The information selected for stems are determined by the category of the stem itself recursively.

Under certain circumstances where a token has two or more parses that agree in the selected features, those parses will be represented by a single projected parse, hence the number of parses in the (projected) training corpus may be smaller than the number of parses in the original corpus. For example, the feature structure above is projected into a feature structure such as:

$$\begin{bmatrix} \text{CAT} & \text{ADJ} \\ \text{STEM} & \begin{bmatrix} \text{CAT} & \text{NOUN} \\ \text{AGR} & \text{3SG} \\ \text{POSS} & \text{1SG} \\ \text{CASE} & \text{LOC} \\ \text{STEM} & \begin{bmatrix} \text{CAT} & \text{VERB} \end{bmatrix} \\ \text{SUFFIX} & \text{DIK} \end{bmatrix} \\ \text{SUFFIX} & \text{REL} \end{bmatrix}$$

3.2 Unknown Words

Although the coverage of our morphological analyzer for Turkish [12], with about 30,000 root words and about 35,000 proper names, is very satisfactory, it is inevitable that there will be forms in the corpora being processed that are not recognized by the morphological analyzer. These are almost always foreign proper names. words adapted into the language and not in the lexicon, or very obscure technical words. These are nevertheless inflected (using Turkish word formation paradigms) with inflectional features demanded by the syntactic context and sometimes even go through derivational processes. For improved disambiguation, one has to at least recover any morphological features even if the root word is unknown. For dealing with this, we have made the assumption that all unknown words have nominal roots, and built a second morphological analyzer whose (nominal) root lexicon recognizes S^+ where S is the Turkish surface alphabet (in the two-level morphology sense) but then tries to interpret an arbitrary postfix of the unknown word as a sequence of Turkish suffixes subject to all morphographemic constraints. For instance when a form such as *talkshowumun* is entered, this second analyzer hypothesizes the following analyses:

```
1. [[CAT NOUN] [ROOT talkshowumun] [AGR 3SG] [POSS NONE] [CASE NOM]]
2. [[CAT NOUN] [ROOT talkshowumu]  [AGR 3SG] [POSS 2SG] [CASE NOM]]
3. [[CAT NOUN] [ROOT talkshowum]   [AGR 3SG] [POSS NONE] [CASE GEN]]
4. [[CAT NOUN] [ROOT talkshowum]   [AGR 3SG] [POSS 2SG] [CASE NOM]]
5. [[CAT NOUN] [ROOT talkshowu]    [AGR 3SG] [POSS 1SG] [CASE GEN]]
6. [[CAT NOUN] [ROOT talkshow]     [AGR 3SG] [POSS 1SG] [CASE GEN]]
```

which are then processed just like any other during disambiguation.[6]

This however is not a sufficient solution for some very obscure situations where for the foreign word is written using its, say, English orthography, while suffixation goes on according to its English pronunciation, which may make some constraints like vowel harmony inapplicable on the graphemic representation, though harmony is in effect in the pronunciation. For instance one sees the form

[6] Incidentally, the correct analysis is the last one meaning *of my talk show*.

Carter'a where the last vowel in *Carter* is pronounced so that it harmonizes with *a* in Turkish, while the *e* in the surface form does not harmonize with *a*. We are nevertheless rather satisfied with our solution as in our experiments we have noted that *well below* 1% of the forms remain as unknown and these are usually item markers in formatted or itemized lists, or obscure foreign acronyms.

3.3 Constraint Rules

The system uses rules of the sort

if `LC` and `RC` then choose `PARSE` or if `LC` and `RC` then delete `PARSE`

where `LC` and `RC` are feature constraints on unambiguous left and right contexts of a given token, and `PARSE` is a feature constraint on the parse(s) that is (are) chosen (or deleted) in that context if there are any parses subsumed by that constraint. Currently the left and right contexts can be at most 2 tokens, hence we look at a window of at most 5 tokens of which one is ambiguous. We refer to the unambiguous tokens in the context as `llc` (left-left context) `lc` (left context), `rc` (right context) and `rrc` (right-right context). Depending on the amount of unambiguous tokens in a context our rules can have one of the following context structures, listed in order of decreasing specificity:

```
1.     llc, lc  ____  rc, rrc

2.     llc, lc  ____
                ____  rc, rrc

3.          lc  ____  rc

4.          lc  ____
                ____  rc
```

To illustrate the flavor of our rules we can give the following examples. The first example chooses parses with case feature ablative, preceding an unambiguous postposition which subcategorizes for an ablative nominal form.

```
[llc:[],lc:[], choose:[case:abl], rc:[[cat:postp,subcat:abl]],rrc:[]]
```

Another sample rule is:

```
[llc:[],lc:[[agr:'2SG',case:gen]],choose:[cat:noun,poss:'2SG'], rc:[],rrc:[]]
```

which chooses a nominal form with a possessive marker `2SG` following an unambiguous (pronomial) form with `2SG` agreement and genitive case, enforcing the simplest form noun–noun, compound noun phrase constraints.

Our system uses two hand-crafted sets of rules, in addition to the rules that are learned by unsupervised learning:

1. We use an initial set of hand-crafted *choose rules* to speed-up the learning process by creating disambiguated contexts over which statistics can be collected. These rules are independent of the corpus that is to be tagged and are linguistically motivated. They contain some very common feature patterns, such as noun–noun, adj–noun, noun–adj–noun, noun–postposition, verb–end-of-sentence constraints. The motivation behind these rules is that they should improve precision without sacrificing recall. *These are rules which impose very tight constraints so as not to make any recall errors.* Our experience is that after processing with these rules, the recall is above 99% while precision improves by about 20 percentage points. *Another important feature of these rules is that they are applied even if the contexts are also ambiguous*, as the constraints are tight. That is, if each token in a sequence of, say, three ambiguous tokens have a parse matching one of the context constraints (in the proper order), then all of them are simultaneously disambiguated. In hand crafting these rules, we have used our experience from an earlier tagger [13].

2. We also use a set of heuristic *delete rules* to get rid of any low probability parses still remaining. For instance, in Turkish, postpositions have rather strict contextual constraints and if there are tokens remaining with multiple parses one of which is a postposition reading, we delete that reading. As stated, these are heuristics and may occasionally delete correct parses. Our experience is that these rules improve precision by about 10 to 12 additional percentage points with negligible impact on recall.

3.4 Learning Choose Rules

Given a training corpus, with tokens annotated with possible parses (projected over selected features), we first apply the initial choose rules. Learning then goes on as a number of iterations over the training corpus. We proceed with following schema which is an adaptation of Brill's formulation [3]:

1. We generate a table, called *incontext*, of all possible unambiguous contexts which contain a token with an unambiguous (projected) parse, along with a count of how many times this parse occurs unambiguously in exactly the same context in the corpus. We refer to an entry in table with a context C and parse P as $incontext(C, P)$.

2. We also generate a table, called *count*, of all unambiguous parses in the corpus along with a count of how many times this parse occurs in the corpus. We refer to an entry in this table with a given parse P, as $count(P)$.

3. We then start going over the corpus token by token generating contexts as we go.

4. For each unambiguous context encountered, $C = (\texttt{LC}, \texttt{RC})$ [7] around an *ambiguous* token w with parses $P_1, \ldots P_k$, and for each parse P_i, we generate a candidate rule of the sort

   if LC and RC then choose P_i

5. Every such candidate rule is then scored in the following fashion:

   (a) We compute

   $$P_{max} = argmax_{P_j\ (j \neq i)}\ \frac{count(P_i)}{count(P_j)} \cdot incontext(C, P_j)$$

   (b) The score of the candidate rule is then computed as

   $$Score_i = incontext(C, P_i) - \frac{count(P_i)}{count(P_{max})} \cdot incontext(C, P_{max})$$

6. We order all candidate rules generated during one pass over the corpus, along two dimensions:

   (a) we group candidate rules by *context specificity* (given by the order in Section 3.3),

   (b) in each group, we order rules by descending *score.*

   We maintain score thresholds associated with each context specificity group: the threshold of a less specific group being higher than that of a more specific group. We then choose the top scoring rule from any group whose score equals or exceeds the threshold associated with that group. The reasoning is that we prefer more specific and/or high scoring rules: high scoring rules are applicable, in general, in more places; while more specific rules have stricter constraints and more accurate morphological parse selections, as we have noted that choosing the highest scoring rule at every step may sometimes make premature commitments which can not be undone later.

7. The selected rules are then applied in the matching contexts and ambiguity in those contexts is reduced. During this application the following are also performed:

   (a) if the application results in an unambiguous parse in the context of the applied rule, we increment the count associated with this parse in table *count*. We also update the *incontext* table for the same context, and other contexts which contains the disambiguated parse.

   (b) we also generate any new unambiguous contexts that this newly disambiguated token may give rise to, and add it to the *incontext* table along with count 1.

   Note that for efficiency reasons rule candidates are not generated repeatedly during each pass over the corpus, but rather once at the beginning, and then when selected rules are applied to very specific portions of the corpus.

8. If there are no rules in any group that exceed its threshold, group thresholds are reduced by multiplying by a damping constant d $(0 < d < 1)$ and iterations are continued.

9. If the threshold for the most specific context falls below a given lower limit, the learning process is terminated.

[7] Either of LC or RC may be empty.

3.5 Learning Delete Rules

The learning process for delete rules is applied after the projected corpus is processed with the initial choose rules, the choose rule learning process, and the hand-crafted delete rules. The operation is essentially as above with some very important differences:

1. context specificity and thresholds are not used,
2. the rules are scored on absolute manner.

For any context that has one ambiguous token still remaining, the remaining (projected) parses are ordered with respect to their frequency of occurrence in the same context in the whole text. A delete rule is then generated for the *lowest scoring parse(s)* and the parse is deleted, with contexts and relevant statistical information updated appropriately afterwards. We iterate until the score of the worst parse exceeds some predefined limit.[8]

3.6 Contexts induced by morphological derivation

The procedure outlined in the previous section has to be modified slightly in the case when the unambiguous token in the `rc` position is a morphologically derived form. For such cases one has to take into consideration additional pieces of information. We will motivate this using a simple example from Turkish. Consider the example fragment:

... bir masa+dır.
... *a* *table+is*
... *is* *a table*

where the first token has the morphological parses:

```
1. [[CAT ADJ] [ROOT bir] [TYPE CARDINAL]]   (one)
2. [[CAT ADJ] [ROOT bir] [TYPE DETERMINER]] (a)
3. [[CAT ADVERB] [ROOT bir]]                (only/merely)
```

and the second form has the unambiguous morphological parse:

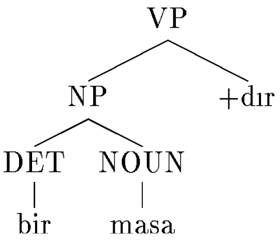
```
1. [[CAT NOUN] [ROOT masa] [AGR 3SG] [POSS NONE] [CASE NOM]
         [CONV VERB NONE] [TAM1 PRES] [AGR 3SG]]  (is table)
```

[8] The current scoring method is not satisfactory for a number of reasons. It tends to generate a large number of rules. We are currently investigating other ways of scoring and selecting delete rules which will be reported in a subsequent paper.

which in hierarchical form corresponds to the feature structure:

$$\begin{bmatrix} \text{CAT} & \text{VERB} \\ \text{TAM1} & \text{PRES} \\ \text{AGR} & \text{3SG} \\ \text{STEM} & \begin{bmatrix} \text{CAT} & \text{NOUN} \\ \text{ROOT} & \text{masa} \\ \text{AGR} & \text{3SG} \\ \text{POSS} & \text{NONE} \\ \text{CASE} & \text{NOM} \end{bmatrix} \\ \text{SUFFIX} & \text{NONE} \end{bmatrix}$$

In the syntactic context this fragment is interpreted as



where the the determiner is attached to the noun and the whole phrase is then taken as a VP although the verbal marker is on the second lexical item. If in this case, the token *bir* is considered to neighbor a token whose top level inflectional features indicate it is a verb, it is likely that *bir* will be chosen as an adverb as it precedes a verb, whereas the correct parse is the determiner reading.

In such a case where the right context of an ambiguous token is a derived form, one has to consider as the right context, both the top level features of final form, and the *stem* from which it was derived. During the set-up of the *incontext* table, such a context is entered twice: once with the top level feature constraints of the immediate unambiguous right-context, and once with the feature constraints of the stem. The unambiguous token in the right context is also entered to the *count* table once with its top level feature structure and once with the feature structure of the stem.

When generating candidate choose or delete rules, for contexts where `rc` is a derived form and `rrc` is empty, we actually generate two candidates rules for each ambiguous token in that context:

1. if `llc, lc` and `rc` then choose/delete P_i.
2. if `llc, lc` and *stem*(`rc`) then choose/delete P_i.

These candidate rules are then evaluated as described above. In general all derivations in a lexical form have to be considered though we have noted that considering one level gives satisfactory results.

3.7 Ignoring Features

Some morphological features are only meaningful or relevant for disambiguation only when they appear to the left or to the right of the token to be disambiguated. For instance, in the case of Turkish, the `CASE` feature of a nominal form is only useful in the immediate left context, while the `POSS` (the possessive agreement marker) is useful only in the right context. If these features along with their possible values are included in context positions where they are not relevant, they "split" scores and hence cause the selection of some other irrelevant rule. Using the maxim that union gives strength, we create contexts so that features not relevant to a context position are not included, thereby treating context that differ in these features as same.[9]

3.8 Applying Learned Rules

Given a new text annotated with all morphological parses (this time the parses are not projected), we apply the rules using the following order:

1. The initial hand-crafted choose rules are applied first. These rules always constrain top level inflectional features, and hence any stems from derivational processes are not considered unless explicitly indicated in the constraint itself.
2. The choose rules that have been learned earlier, are then repeatedly applied to unambiguous contexts, until no more ambiguity reduction is possible. During the application of these rules, if the immediate right context of a token is a derived form, then the stem of the right context is also checked against the constraint imposed by the rule. So if the rule right context constraint subsumes the top level feature structure or the stem feature structure, then the rule succeeds and is applied if all other constraints are also satisfied.
3. The hand-crafted delete clean-up rules are applied to any remaining parses.
4. Finally, the delete rules that have been learned are applied repeatedly to unambiguous contexts, until no more ambiguity reduction is possible.

4 Experimental Results

We have applied our learning system to several Turkish texts. Some statistics on these texts are given in Table 1. Here, the tokens considered are after what is generated after morphological analysis, unknown word processing and any lexical coalescing is done. The words that are unknown are those that could not even be processed by the unknown noun processor. Whenever an unknown word had more than one parse it was counted under the appropriate group.

[9] Obviously these features are specific to a language.

Text	Sentences	Tokens	Distribution of Morphological Parses					
			0	1	2	3	4	> 4
ARK	492	7,928	0.15%	49.34%	30.93%	9.19%	8.46%	1.93%
C2400	2,407	39,800	0.03%	50.56%	28.66%	10.12%	8.16%	2.47%
C270	270	5212	0.02%	50.63%	30.68%	8.62%	8.36&	1.69%

Table 1: Statistics on Texts

As the first text that we experimented with was rather small, we essentially learned using this text and then applied the initial, learned and final heuristic rules on it directly. For the second text we trained the tagger on the first 500, 1000 portions of the second text, and applied these to a previously unseen text C270 which was set aside for testing. Gold standard disambiguated versions for ARK and C270 were prepared manually to evaluate the automatically tagged versions.

Our results are summarized in the following set of tables. Table 2 gives the number of choose rules in the set of rules learned during training with different stopping score thresholds. There were 304 rules in the initial set of hand-crafted rules and 41 rules in the final set of heuristic rules. In Table2, the columns under *VAR* indicate the number of rules induced using the regime presented before and the columns under *MAX* indicate the number of rules induced by selecting the rule that score the highest as done by Brill [3].

Stopping Score	Training Texts					
	ARK		C500		C1000	
	VAR	MAX	VAR	MAX	VAR	MAX
1.2	180	546	165	604	361	1213
3.0	50	257	52	362	110	773
5.0	26	167	30	197	56	444
7.0	19	113	15	129	43	290
14.0	6	61	10	65	14	131
28.0	0	17	0	25	10	60

C500, C1000 are the initial 500, 1000 sentence portions of text C2400.

Table 2: Number of additional constraints learned from various training texts.

We then tagged ARK and C270 using only the initial choose rules, and the initial choose rules followed by the final delete rules, to see where we would be without any learning component. We present these results in rows 2 and 3 of Tables 3 and 4, titled CHOOSE and CHOOSE + DELETE. The first row titled BASE gives the statistics on the unprocessed text. Please note that for ARK the test text is the same as the training text, while for C270, the training texts are different from the test text.

Tagging Experiment	Parses/ Token	Recall (%)	Precision (%)
BASE	1.828	100.00	54.69
CHOOSE ONLY	1.310	99.36	75.84
CHOOSE + DELETE	1.134	99.30	87.50
FULL	1.051	96.76	92.04

Table 3: Average parses, recall and precision for text ARK

Tagging Experiment	Parses/ Token	Recall (%)	Precision (%)
BASE	1.719	100.00	58.18
CHOOSE ONLY	1.342	99.19	73.87
CHOOSE + DELETE	1.167	98.81	84.68
FULL-500	1.107	97.06	87.78
FULL-1000	1.100	96.99	88.13

Table 4: Average parses, recall and precision for text C270

Figures 2 and 3 show the results of applying initial choose rules, choose rules learned with different stopping thresholds, and hand-crafted delete rules. We have then learned the delete rules following the learning of one of the better scoring choose rules. The test text was then processed with the initial choose rules, the learned choose rules, the delete rules and the learned delete rules. The numerical results for these are in the fourth column of Tables 3 and 4, titled FULL.

4.1 Discussion of Results

Our results indicate that, it is possible to use unsupervised learning in a constraint-based morphological disambiguation system. Hand-crafted rules go a long way in improving precision substantially, but in a language like Turkish, one has to code rules that allow no, or only carefully controlled derivations, otherwise lots of things go massively wrong. Thus we have used very tight and conservative rules in hand-crafting. Learning choose rules then produces additional rules some of which look very much like the hand-crafted rules. An important class of rules we explicitly avoid hand crafting are the rules for disambiguating around coordinating conjunctions. We have noted that while learning choose rules, the system zeroes in rather quickly on these contexts and comes up with rather successful rules for conjunctions. The learning process for delete rules finds rather interesting rules for resolving some notoriously difficult cases such the ambiguity between accusative and nominative case nominal forms, with no and 3SG possessive marker respectively, as the surface forms for both cases are the same for roots ending in a consonant.

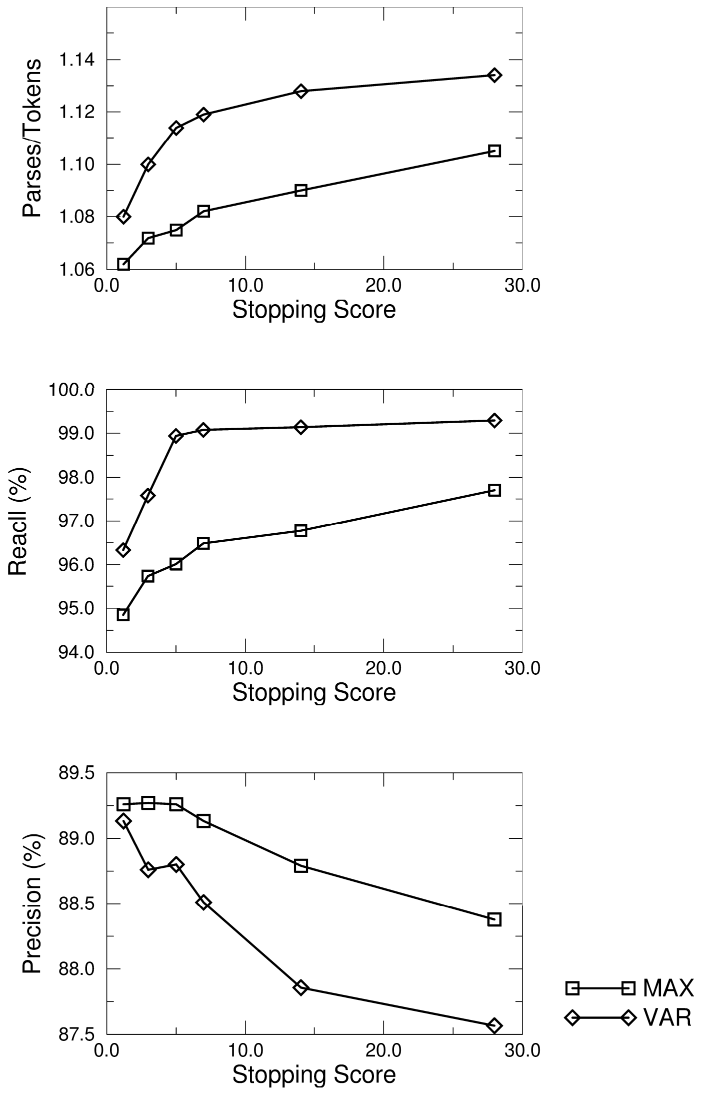


Figure 2: Average parses, recall and precision plots for text ARK.

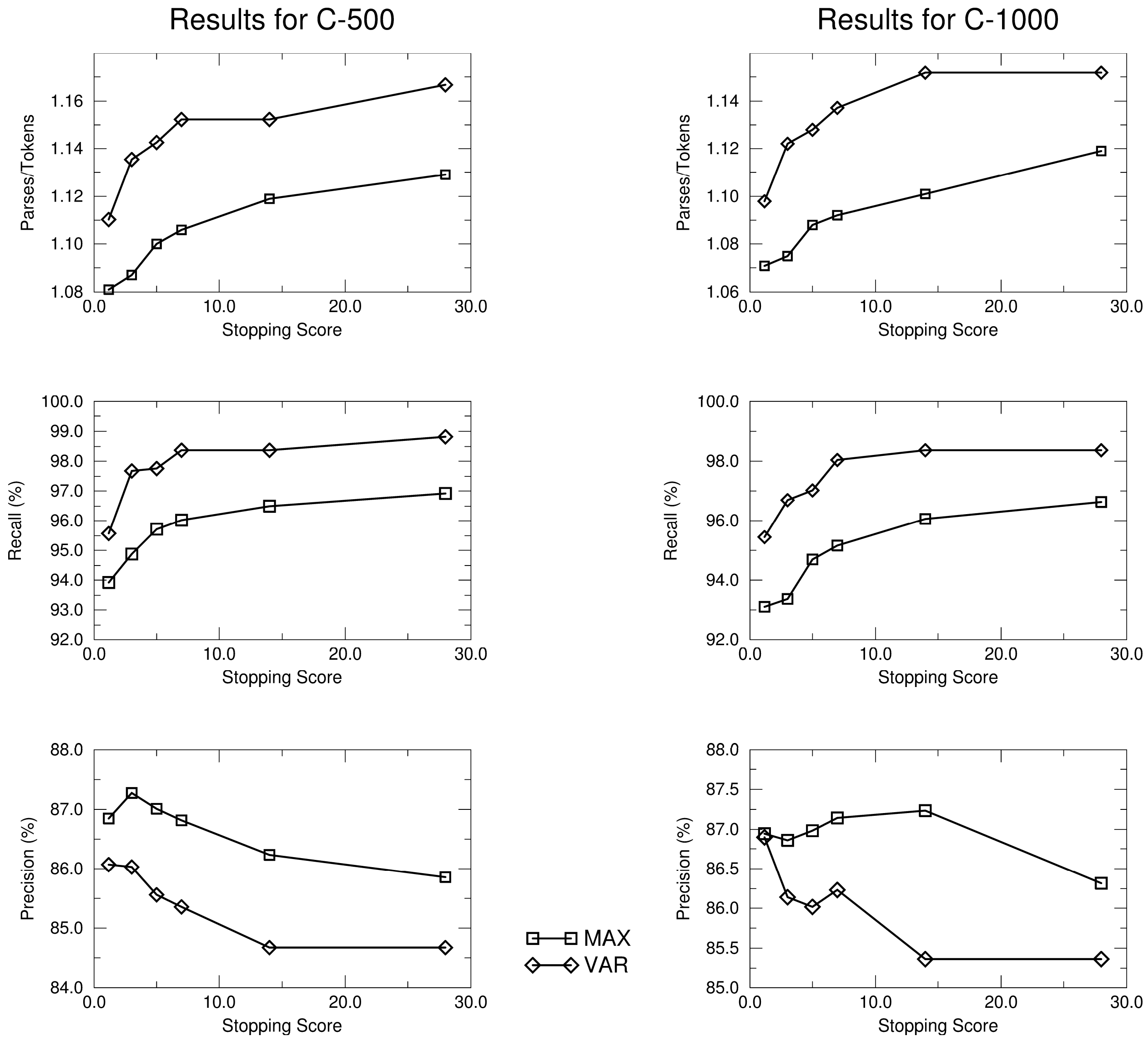


Figure 3: Average parses, recall and precision plots for text C270 using different learning texts.

Our results indicate the obvious trade-off between recall and precision. During the learning of the choose rules, the VAR regime for selecting rules gives consistently better recall results and consistently lower precision results compared to the MAX regime. One obvious disadvantage of the MAX regime is that it consistently generates much larger number of rules, but there is no proportionally better returns from larger number of rules. Thus, depending on whether recall or precision is more important for the resulting text, one has the option of using the VAR or the MAX regime and/or different stopping thresholds. Our use of delete rules in a learning process is novel, but we feel that there is some more room here for improvement as we are not very pleased with our scoring. We are also not very pleased with the way the stopping threshold is used and are looking into ways of making it independent of the corpus size.

Our system has been implemented in Prolog and runs on Sun SparcStations. On a Sparc 10, it takes about 10 minutes to learn choose rules on the ARK text. Tagging is slightly better. We are however porting our system to Sicstus Prolog 3.0 which can generate native Sun code so we expect improvements in time performance.

From analysis of our results we have noted that trying to choose one correct parse for every token is rather ambitious (at least for Turkish). There are a number of reasons for this:

- A given word may be interpreted in more than one way but with the same inflectional features, or with features not inconsistent with the syntactic context. This usually happens when the root of one of the forms is a proper prefix of the root of the other one. One would need serious amounts of semantic, or statistical root word and word form preference information for resolving these. For instance, in

...	koyun	sürüsü	...
...	koyun	sürü+sü	...
...	sheep	herd+POSS-3SG	(sheep herd)
...	koy+un	sürü+sü	...
...	bay+GEN	herd+POSS-3SG	(?? bay's herd)

  both noun phrases are syntactically possible, though the second one is obviously nonsense. It is not clear how one would disambiguate this using just syntactic information.

  Another similar example is:

...	kurmaya	yardım	etti	....
...	kur+ma+ya	yardım	et+ti	...
...	construct+INF+DAT	help	make+PAST	helped construct (something)
...	kurmay+a	yardım	et+ti	...
...	military-officer+DAT	help	make+PAST	helped the military-officer

  where again with have a similar problem. It may be possible to resolve this one using subcategorization constraints on the object of the verb *kur* assuming it is in the very near preceding context, but this may be very unlikely as Turkish allows arbitrary adjuncts between the object and the verb.

- Turkish allows sentences to consist of a number of sentences separated by commas. Hence locating a verb in the middle of a sentence is rather difficult, as certain verbal forms also have an adjectival reading, and punctuation is not very helpful as commas have many other uses.

- The distance between two constituents (of, say, a noun phrase) that have to agree in various morphosyntactic features may be arbitrarily long and this causes occasional mismatches, especially if the right nominal constituent has a surface plural marker which causes a 4-way ambiguity, as in for *masaları*.

```
     masalarI
  1. [[CAT NOUN] [ROOT masa] [AGR 3PL] [POSS NONE] [CASE ACC]] (tables accusative)
  2. [[CAT NOUN] [ROOT masa] [AGR 3PL] [POSS 3SG] [CASE NOM]]  (his tables)
  3. [[CAT NOUN] [ROOT masa] [AGR 3PL] [POSS 3PL] [CASE NOM]]  (their tables)
  4. [[CAT NOUN] [ROOT masa] [AGR 3SG] [POSS 3PL] [CASE NOM]]  (their table)
```

  Choosing among the last three is rather problematic if the corresponding genitive form is outside the context.

Among these problems, the most crucial is the first one which we believe can be solved to a great extent by using root word preference statistics and word form preference statistics as used by Levinger *et al.* [11].

5 Conclusions

This paper has presented a rule-based morphological disambiguation approach which uses a set of hand-crafted constraint rules and learns additional rules to choose and delete parses, from untagged text in an unsupervised manner. We have extended the rule learning and application schemes so that the impact of various morphological phenomena and features are selectively taken into account. We have applied our approach to the morphological disambiguation of Turkish, a free–constituent order language, with agglutinative morphology, exhibiting productive inflectional and derivational processes. We have also incorporated a rather sophisticated unknown form processor which extracts any relevant inflectional or derivational markers even if the root word is unknown.

Our results indicate that using hand-crafted rules and rules learned to choose, we can attain a recall of 99.08% and a precision of 88.08% with 1.119 parses per token, on the training text. When rules learned to delete are used in addition to these, we can attain a recall of 96.76% and a precision of 92.05% and 1.051 parses per token on the training text. On previously unseen text, we can attain a recall of 98.04% and a precision of 86.23% with 1.137 parses per token using just the hand-crafted rules and rules learned to choose. When rules learned to delete are used we can attain a recall of 96.99% and a precision of 88.13% and 1.100 parses per token.

In addition to improving our approach to the learning of delete rules, we are also investigating the use of root word preference statistics that we can obtain from tagged texts.

6 Acknowledgments

We would like to thank Xerox Advanced Document Systems, and Lauri Karttunen of Xerox Parc and of Rank Xerox Research Centre (Grenoble) for providing us with the two-level transducer development software on which the morphological and unknown word recognizer were implemented. This research has been supported in part by a NATO Science for Stability Grant TU–LANGUAGE.

References

[1] E. Brill. A simple-rule based part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[2] E. Brill. Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Articial Intelligence (AAAI-94)*, Seattle, Washinton, 1994.

[3] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, MA, June 1995.

[4] K. W. Church. A stochastic parts program and a noun phrase parser for unrestricted text. In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas, 1988.

[5] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy, 1992.

[6] S. J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, 1988.

[7] Z. Güngördü and K. Oflazer. Parsing Turkish using the Lexical-Functional Grammar formalism. *Machine Translation*, to appear in 1996.

[8] F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Anttila. *Constraint Grammar-A Language–Independent System for Parsing Unrestricted Text.* Mouton de Gruyter, 1995.

[9] L. Karttunen. Finite-state lexicon compiler. XEROX, Palo Alto Research Center– Technical Report, April 1993.

[10] İ. Kuruöz. Tagging and morphological disambiguation of Turkish text. Master's thesis, Bilkent University, Department of Computer Engineering and Information Science, July 1994.

[11] M. Levinger, U. Ornan, and A. Itai. Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21(3):383–404, September 1995.

[12] K. Oflazer. Two-level description of Turkish morphology. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, April 1993. A full version appears in *Literary and Linguistic Computing*, Vol.9 No.2, 1994.

[13] K. Oflazer and İ. Kuruöz. Tagging and morphological disambiguation of Turkish text. In *Proceedings of the* 4^{th} *Applied Natural Language Processing Conference*, pages 144–149. ACL, October 1994.

[14] A. Voutilainen. Morphological disambiguation. In F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Anttila, editors, *Constraint Grammar-A Language–Independent System for Parsing Unrestricted Text*, chapter 5. Mouton de Gruyter, 1995.

[15] A. Voutilainen. A syntax-based part-of-speech analyzer. In *Proceedings of the Seventh Conference of the European Chapter of the Association of Computational Linguistics*, Dublin, Ireland, 1995.

[16] A. Voutilainen, J. Heikkilä, and A. Anttila. *Constraint Grammar of English.* University of Helsinki, 1992.

[17] A. Voutilainen and P. Tapanainen. Ambiguity resolution in a reductionistic parser. In *Proceedings of EACL'93*, Utrecht, Holland, 1993.