Morphological Disambiguation by Voting Constraints

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

Abstract
We present a constraint-based morphological disambiguation system in which individual constraints vote on matching morphological parses, and disambiguation of all the tokens in a sentence is performed at the end by selecting parses that receive the highest votes. This constraint application paradigm makes the outcome of the disambiguation independent of the rule sequence, and hence relieves the rule developer from worrying about potentially conflicting rule sequencing. Our results for disambiguating Turkish indicate that using about 500 constraint rules and some additional simple statistics, we can attain a recall of 95-96% and a precision of 94-95% with about 1.01 parses per token. Our system is implemented in Prolog and we are currently investigating an efficient implementation based on finite state transducers.

1 Introduction
Automatic morphological disambiguation is an important component in higher level analysis of natural language text corpora. There has been a large number of studies in tagging and morphological disambiguation using various techniques such as statistical techniques, e.g., (Church, 1988; Cutting et al., 1992; DeRose, 1988), constraint-based techniques (Karlsson et al., 1995; Voutilainen, 1995b; Voutilainen, Heikkilä, and Anttila, 1992; Voutilainen and Tapanainen, 1993; Oflazer and Kuruöz, 1994; Oflazer and Tür, 1996) and transformation-based techniques (Brill, 1992; Brill, 1994; Brill, 1995).

This paper presents a novel approach to constraint based morphological disambiguation which relieves the rule developer from worrying about conflicting rule ordering requirements. The approach depends on assigning votes to constraints according to their complexity and specificity, and then letting constraints cast votes on matching parses of a given lexical item. This approach does not reflect the outcome of matching constraints to the set of morphological parses immediately. Only after all applicable rules are applied to a sentence, all tokens are disambiguated in parallel. Thus, the outcome of the rule applications is independent of the order of rule applications. Rule ordering issue has been discussed by Voutilainen(1994), but he has recently indicated\(^1\) that insensitivity to rule ordering is not a property of their system (although Voutilainen(1995a) states that it is a very desirable property) but rather is achieved by extensively testing and tuning the rules.

In the following sections, we present an overview of the morphological disambiguation problem, highlighted with examples from Turkish. We then present our approach and results. We finally conclude with a very brief outline of our investigation into efficient implementations of our approach.

2 Morphological Disambiguation
In all languages, words are usually ambiguous in their parts-of-speech or other morphological features, and may represent lexical items of different syntactic categories, or morphological structures depending on the syntactic and semantic context. In languages like English, there are a very small number of possible word forms that can be generated from a given root word, and a small number of part-of-speech tags associated with a given lexical form. On the other hand, in languages like Turkish or Finnish with very productive agglutinative morphology, it is possible to produce thousands of forms (or even millions (Hankamer, 1989)) from a given root word and the kinds of ambiguities one observes are quite different than what is observed in languages like English.

In Turkish, there are ambiguities of the sort typically found in languages like English (e.g., book/noun vs book/verb type). However, the agglutinative nature of the language usually helps resolution of such ambiguities due to the restrictions on morphotactics of subsequent morphemes. On the

\(^1\)Voutilainen, Private communication.
other hand, this very nature introduces another kind of ambiguity, where a lexical form can be morphologically interpreted in many ways not usually predictable in advance. Furthermore, Turkish allows very productive derivational processes and the information about the derivational structure of a word form is usually crucial for disambiguation (Oflazer and Tür, 1996).

Most kinds of morphological ambiguities that we have observed in Turkish typically fall into one of the following classes:

1. The form is uninflected and assumes the default inflectional features, e.g.,

   | CAT | ROOT | AGR | POSS | CASE |
   |-----|------|-----|------|------|
   | ADJ | taS  | 3SG | NONE | NOM  |

2. Lexically different affixes (conveying different morphological features) surface the same due to the morphographemic context, e.g.,

   | CAT | ROOT | AGR | POSS | CASE |
   |-----|------|-----|------|------|
   | NOUN| ev+u | 3SG | NONE | GEN  |

3. The root of one of the parses is a prefix string of the root of the other parse, and the parse with the shorter root word has a suffix which surfaces as the rest of the longer root word, e.g.,

   | CAT | ROOT | AGR | POSS | CASE |
   |-----|------|-----|------|------|
   | ADJ | koyu| 3SG | NONE | NOM  |

4. The roots take different numbers of unrelated inflectional and/or derivational suffixes which when concatenated turn out to have the same surface form, e.g.,

   | CAT | ROOT | AGR | POSS | CASE |
   |-----|------|-----|------|------|
   | VERB| yap | 2PL | NONE | ABL  |

5. The main intent of our system is to achieve morphological disambiguation by choosing for a given ambiguous token, the correct parse in a given context. 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, but one will be the "correct" parse usually on semantic grounds. We consider a token fully disambiguated if it has only one morphological parse remaining after automatic disambiguation. We consider a 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 (Voutilainen, 1995a):

\[ \text{Ambiguity} = \frac{\#\text{Parses}}{\#\text{Tokens}} \]
\[ \text{Recall} = \frac{\#\text{Tokens Correctly Disambiguated}}{\#\text{Tokens}} \]
\[ \text{Precision} = \frac{\#\text{Tokens Correctly Disambiguated}}{\#\text{Parses}} \]

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,\(^3\) 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-based Morphological Disambiguation

This section outlines our approach to constraint-based morphological disambiguation where con-
straints vote on matching parses of sequential to-
kens.

3.1 Constraints on morphological parses

We describe constraints on the morphological parses
of tokens using rules with two components
\[ R = (C_1, C_2, \ldots, C_n; V) \]
where the \( C_i \) are (possibly hierarchical) feature con-
straints on a sequence of the morphological parses,
and \( V \) is an integer denoting the vote of the rule.

To illustrate the flavor of our rules we can give the
following examples:

1. The following rule with two constraints matches
    parses with case feature ablative, preceding a
    parse matching a postposition subcategorizing
    for an ablative nominal form.
    \[ ([\text{case}: \text{abl}], [\text{cat}: \text{postp}, \text{subcat}: \text{abl}]) \]

2. The rule
    \[ ([\text{agr}: '2SG', \text{case}: \text{gen}], [\text{cat}: \text{noun}, \text{poss}: '2SG']) \]
    matches a nominal form with a possessive
    marker 2SG, following a pronoun with 2SG
    agreement and genitive case, enforcing the sim-
    plest form of noun phrase constraints.

3. In general constraints can make references to
    the derivational structure of the lexical form
    and hence be hierarchical. For instance, the fol-
    lowing rule is an example of a rule employing a
    hierarchical constraint:
    \[ ([\text{cat}: \text{adj}, \text{stem}: [\text{cat}: \text{v}], \text{suffix}: \text{mis}]),
       ([\text{cat}: \text{noun}, \text{stem}: \text{no}]) \]
    which matches the derived participle reading of
    a verb with narrative past tense, if it is followed
    by an underived noun parse.

3.2 Determining the vote of a rule

There are a number of ways votes can be assigned
to rules. For the purposes of this work the vote of
a rule is determined by its static properties, but it
is certainly conceivable that votes can be assigned
or learned by using statistics from disambiguated
corpora.\(^4\) For static vote assignment, intuitively, we
would like to give high votes to rules that are more
specific: i.e., to rules that have

- higher number of constraints,
- higher number of features in the constraints,
- constraints that make reference to nested stems
  (from which the current form is derived),
- constraints that make reference to very specific
  features or values.

Let \( R = (C_1, C_2, \ldots, C_n; V) \) be a constraint rule.
The vote \( V \) is determined as
\[
V = \sum_{i=1}^{n} V(C_i)
\]
where \( V(C_i) \) is the contribution of constraint \( C_i \)
to the vote of the rule \( R \). A (generic) constraint has
the following form:
\[
C = [(f_1 : v_1)(f_2 : v_2)\&\cdots(f_m : v_m)]
\]
where \( f_i \) is the name of a morphological feature, and
\( v_i \) is one of the possible values for that feature. The
contribution of \( f_i : v_i \) in the vote of a constraint
depends on a number of factors:

1. The value \( v_i \) may be a distinguished value that
    has a more important function in disambugua-
tion.\(^5\) In this case, the weight of the feature
    constraint is \( w(v_i)(>1) \).
2. The feature itself may be a distinguished feature
    which has more important function in disam-
    biguation. In this case the weight of the feature
    is \( w(f_i)(>1) \).
3. If the feature \( f_i \) refers to the stem of a de-
    rived form and the value part of the feature con-
    straint is a full fledged constraint \( C' \) on the stem
    structure, the weight of the feature constraint is
    found by recursively computing the vote of \( C' \)
and scaling the resulting value by a factor (2 in
our current system) to improve its specificity.
4. Otherwise, the weight of the feature constraint
    is 1.

For example suppose we have the following con-
straint:
\[ [\text{cat}: \text{noun}, \text{case}: \text{gen},
    \text{stem}: [\text{cat}: \text{adj}, \text{stem}: [\text{cat}: \text{v}], \text{suffix}: \text{mis}]] \]
Assuming the value \text{gen} is a distinguished value
with weight 4 (cf., factor 1 above), the vote of this
constraint is computed as follows:

1. \text{cat}: \text{noun} contributes 1,
2. \text{case}: \text{gen} contributes 4,
3. \text{stem}: [\text{cat}: \text{adj}, \text{stem}: [\text{cat}: \text{v}], \text{suffix}: \text{mis}]
    contributes 8 computed as follows:
    (a) \text{cat}: \text{adj} contributes 1,
\[^3\text{Assuming no unknown words.}
\[^4\text{We have left this for future work.}
\[^5\text{For instance, for Turkish we have noted that the genitive case marker is usually very helpful in disambiguation.} \]
of the text one is disambiguating. For instance if one

The weights for these rules are currently manually
terences among the parses of single lexical form in-

ditions in a sentence and votes are tallied, morpho-
much than one parse of a token, then the votes of all
straints match, the votes of all the matching parses
sequence of tokens $w_i, w_{i+1}, \cdots, w_{i+n-1}$ within a sen-
A rule $R = (C_1, C_2, \cdots, C_n; V)$ will match a se-
are discouraged.

4 Votes from steps 1, 2 and 3(d) are added up to
give 13 as the constraint vote.

We also employ a set of rules which express prefer-
ences among the parses of single lexical form inde-

ten sequence of tokens $w_i$ through $w_j$, if some morphological parse of
every token $w_j, i \leq j \leq i + n - 1$ is subsumed by
the corresponding constraint $C_j; i+1$. When all con-
straints match, the votes of all the matching parses
are incremented by $V$. If a given constraint matches
more than one parse of a token, then the votes of all
such matching parses are incremented.

After all rules have been applied to all token po-
sitions in a sentence and votes are tallied, morpho-

4 Results from Disambiguating

Turkish Text

We have applied our approach to disambiguating
Turkish text. Raw text is processed by a prepro-
cessor which segments the text into sentences using

dents that span a context much wider that 5

dar precision for the $m = 1$ causes a dramatic increase in
precision for those cases. Even at $m = 0.95$
there is considerable loss of precision and going up
to $m = 1$ causes a dramatic increase in precision
without a significant loss in recall. It can be seen
that we can attain very good recall and quite ac-
table precision with just voting constraint rules.

Our experience is that we can in principle add highly
specialized rules by covering a larger text base to
improve our recall and precision for the $m = 1$. A
post-mortem analysis has shown that cases that have
been missed are mostly due to morphosyntactic de-
pendencies that span a context much wider that 5
tokens that we currently employ.

4.1 Using root and contextual statistics

We have employed two additional sources of informa-
tion: root word usage statistics, and contextual
statistics. We have statistics compiled from previ-
ously disambiguated text, on root frequencies. After
the application of constraints as described above, for

(b) $\text{suffix=mis}$ contributes 1,
(c) $\text{stem:cat:v}$ contributes $2 = 2 + 1$, the 1
being from $\text{cat:v}$,
(d) the sum 4 is scaled by 2 to give 8.

4. Voting and selecting parses

A rule $R = (C_1, C_2, \cdots, C_n; V)$ will match a se-
quence of tokens $w_i, w_{i+1}, \cdots, w_{i+n-1}$ within a sen-
tence $w_i$ through $w_j$, if some morphological parse of
every token $w_j, i \leq j \leq i + n - 1$ is subsumed by
the corresponding constraint $C_j; i+1$. When all con-
straints match, the votes of all the matching parses
are incremented by $V$. If a given constraint matches
more than one parse of a token, then the votes of all
such matching parses are incremented.

After all rules have been applied to all token po-
sitions in a sentence and votes are tallied, morpho-

4 Results from Disambiguating

Turkish Text

We have applied our approach to disambiguating
Turkish text. Raw text is processed by a prepro-
cessor which segments the text into sentences using

Our experience is that we can in principle add highly
specialized rules by covering a larger text base to
improve our recall and precision for the $m = 1$. A
post-mortem analysis has shown that cases that have
been missed are mostly due to morphosyntactic de-
pendencies that span a context much wider that 5
tokens that we currently employ.

4.1 Using root and contextual statistics

We have employed two additional sources of informa-
tion: root word usage statistics, and contextual
statistics. We have statistics compiled from previ-
ously disambiguated text, on root frequencies. After
the application of constraints as described above, for

The reason for the (comparatively) high number of
unknown words in MAN, is that tokens found in such
texts, like H10, denoting a function key in the computer
can not be parsed as a Turkish root word!
tokens which are still ambiguous with ambiguity resulting from different root words, we discard parses if the frequencies of the root words for those parses are considerably lower than the frequency of the root of the highest scoring parse. The results after applying this step on top of voting, with \( m = 1 \), are shown in the fourth column of Table 3 (labeled V+R).

On top of this, we use the following heuristic using context statistics to eliminate any further ambiguities. For every remaining ambiguous token with unambiguous immediate left and right contexts (i.e., the tokens in the immediate left and right are unambiguous), we perform the following, by ignoring the root/stem feature of the parses:

1. For every ambiguous parse, we count how many times, this parse occurs unambiguously in exactly the same unambiguous context, in the rest of the text.

2. We then choose the parse whose count is substantially higher than the others.

The results after applying this step on the previous two steps are shown in the last column of Table 3 (labeled V+R+C). One can see from the last three columns of this table, the impact of each of the steps.

By ignoring root/stem features during this process, we essentially are considering just the top level inflectional information of the parses. This is very similar to Brill's use of contexts to induce transformation rules for his tagger (Brill, 1992; Brill, 1995), but instead of generating transformation rules from a training text, we gather statistics and apply them to parses in the text being disambiguated.

### 5 Efficient Implementation

Techniques and Extensions

The current implementation of the voting approach is meant to be a proof of concept implementation and is rather inefficient. However, the use of regular relations and finite state transducers (Kaplan and Kay, 1994) provide a very efficient implementation method. For this, we view the parses of the tokens making up a sentence as making up a cyclic finite state recognizer with the states marking word boundaries and the ambiguous interpretations of the tokens as the state transitions between states, as depicted in Figure 1 for a sentence with 5 tokens. In Figure 1, the transition labels are triples of the sort \( (w_i, p_j, 0) \) for the \( j^{th} \) parse of token \( i \), with 0 indicating the initial vote of the parse. The rules imposing constraints can also be represented as transducers which increment the votes of the matching transi-
tion labels by an appropriate amount. Such transducers ignore and pass through unchanged, parses that are not sensitive to.

When a finite state recognizer corresponding to the input sentence (which actually may be considered as an identity transducer) is composed with a constraint transducer, one gets a slightly modified version of the sentence transducer with possibly additional transitions and states, where the votes of some of the labels have been appropriately incremented. When the sentence transducer is composed with all the constraint transducers in sequence, all possible votes are cast and the final sentence transducer reflects all the votes. The parse corresponding to each token with the highest vote can then be selected. The key point here is that due to the nature of the composition operator, the constraint transducers can be composed off-line first, giving a single constraint transducer and then this one is composed with every sentence transducer once (See Figure 2).

The idea of voting can further be extended to a path voting framework where rules vote on paths containing sequences of matching parses and the path from the start state to the final state with the highest votes received, is then selected. This can be implemented again using finite state transducers as described above (except that path vote is apportioned equally to relevant parse votes), but instead of selecting highest scoring parses, one selects the path from the start state to one of the final states where the sum of the parse votes is maximum. We have recently completed a prototype implementation of this approach (in C) for English (Brown Corpus) and have obtained quite similar results (Tüür, Oflazer, and Öz-kan, 1997).

6 Conclusions

We have presented an approach to constraint-based morphological disambiguation which uses constraint voting as its primary mechanism for parse selection and alleviates the rule developer from worrying about rule ordering issues. Our approach is quite general and is applicable to any language. Rules describing language specific linguistic constraints vote on matching parses of tokens, and at the end, parses for every token receiving the highest tokens are selected. We have applied this approach to Turkish, a language with complex agglutinative word forms exhibiting morphological ambiguity phenomena not usually found in languages like English and have obtained quite promising results. The convenience of adding new rules in without worrying about where exactly it goes in terms of rule ordering (something that hampered our progress in our earlier work on disambiguating Turkish morphology (Oflazer and Küruçoğlu, 1994; Oflazer and Tüür, 1996)), has also been a key positive point. Furthermore, it is also possible to use rules with negative votes to disallow impossible cases. This has been quite useful for our work on tagging English (Tüür, Oflazer, and Öz-kan, 1997) where such rules with negative weights were used to fine tune the behavior of the tagger in various problematic cases.

The proposed approach is also amenable to an efficient implementation by finite state transducers (Kaplan and Kay, 1994). By using finite state transducers, it is furthermore possible to use a bit more expressive rule formalism including for instance the Kleene * operator so that one can use a much smaller set of rules to cover the same set of local linguistic phenomena.

Our current and future work in this framework involves the learning of constraints and their votes from corpora, and combining learned and hand-crafted rules.

7 Acknowledgments

This research has been supported in part by a NATO Science for Stability Grant TU–LANGUAGE. We thank Lauri Karttunen of Xerox Research Centre in Grenoble for providing the Xerox two-level morphology tools on which the Turkish morphological analyzer was built.

References

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

Brill, Eric. 1994. Some advances in rule-based part of speech tagging. In Proceedings of the
The composition of the sentence transducer with the constraint transducer is a single transducer composed from all constraint transducers. Figure 2: Sentence and Constraint Transducers
Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics, 21*(1):543–566, December.

Church, Kenneth W. 1988. 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.

Cutting, Doug, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, Trento, Italy.

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

Hankamer, Jorge. 1989. Morphological parsing and the lexicon. In W. Marslen-Wilson, editor, *Lexical Representation and Process*. MIT Press.

Kaplan, Ronald M. and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics, 20*(3):331–378, September.

Karlsson, Fred, Atro Voutilainen, Juha Heikkilä, and Arto Anttila. 1995. *Constraint Grammar-A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Karttunen, Lauri. 1993. Finite-state lexicon compiler. XEROX, Palo Alto Research Center—Technical Report, April.

Oflazer, Kemal and İlíker Kurúöz. 1994. Tagging and morphological disambiguation of Turkish text. In *Proceedings of the 4th Applied Natural Language Processing Conference*, pages 144–149. ACL, October.

Oflazer, Kemal and Gökhan Tüür. 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. In Eric Brill and Kenneth Church, editors, *Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing*.

Tüür, Gökhan, Kemal Oflazer, and Nihat Özkan. 1997. Tagging English by path voting constraints. Technical Report BU-CEIS-9704, Bilkent University, Department of Computer Engineering and Information Science, Ankara, Turkey, March. Available as ftp://ftp.cs.bilkent.edu.tr/pub/tech-reports/1997/BU-CEIS-9704.ps.Z.

Voutilainen, Atro. 1995a. Morphological disambiguation. In Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors, *Constraint Grammar-A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, chapter 5.

Voutilainen, Atro. 1995b. 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.

Voutilainen, Atro, Juha Heikkilä, and Arto Anttila. 1992. *Constraint Grammar of English*. University of Helsinki.

Voutilainen, Atro and Pasi Tapanainen. 1993. Ambiguity resolution in a reductionistic parser. In *Proceedings of EACL‘93*, Utrecht, Holland.