Morphological Disambiguation by Voting Constraints

Kemal Oflazer and **Gökhan Tür**
Department of Computer Engineering and Information Science
Bilkent University, Bilkent, Ankara, 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 very 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 found in other systems. The vote of each rule is determined by its complexity measured with the kind and number of features used in the rule. We have applied our approach in a system for morphological disambiguation of Turkish, a language with complex agglutinative word structures, displaying rather different types of morphological ambiguity not found in languages like English. Our results 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 discrimination networks used in AI production systems

1 Introduction

Automatic morphological disambiguation is a crucial component in higher level analysis of natural language text corpora. Morphological disambiguation also facilitates parsing, essentially by performing a certain amount of ambiguity resolution using relatively cheaper methods. 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 (1988), Cutting et al. (1992), DeRose (1988)). Rule-based or constraint-based approaches recently most prominently exemplified by the Constraint Grammar work (Karlsson et al., 1995; Voutilainen, 1995b; Voutilainen, Heikkilä, and Anttila, 1992; Voutilainen and Tapanainen, 1993), employ 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 (1992; 1994; 1995) has presented a transformation-based learning approach, which induces tagging rules from tagged corpora.

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 (or even millions (Hankamer, 1989)) from a given root word, pose a challenging problem for morphological disambiguation. Our prior attempts in developing constraint-based disambiguation systems for Turkish have been hampered to a certain extent by the idiosyncrasies of rule ordering whereby minor changes to the structure and/or ordering of the rules caused massive breakdowns in performance.

This paper presents a novel approach to constraint based morphological disambiguation which relieves the rule developer from worrying about conflicting rule ordering requirements and constraints. The approach depends on assigning weights to constraints according to their complexity and specificity, and then letting constraints vote 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 not dependent on 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 in 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 an outline of our investigation into efficient implementations of our approach.

2 Morphological Disambiguation

In almost 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. Part-of-speech (POS) tagging involves assigning every word its proper part-of-speech based upon the context the word appears in.

In Turkish, there are ambiguities of the sort typically found in languages like English (e.g., book/N vs book/V type). 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 lexical form can be morphologically interpreted in many ways not usually predictable in advance.

Most kinds of morphological ambiguities that we have observed in Turkish typically fall into one the following classes:[2,3,4]

1. the form is uninflected and assumes the default morphological features. For instance `taş` has the parses:

```
1. taS    (made of stone)
   [[CAT=ADJ][ROOT=taS]]

2. taS    (stone)
    [[CAT=NOUN][ROOT=taS]
      [AGR=3SG][POSS=NONE][CASE=NOM]]

3. taS    (overflow!)
    [CAT=VERB][ROOT=taS]
      [SENSE=POS][TAM1=IMP][AGR=2SG]]
```

2. All parses are inflected, and lexically different affixes surface the same due to the morphophonemic context giving rise to different morphological features in different parses. For instance `evin` has the following parses:

```
1.  ev+[n]in  (of the house)
     [[CAT=NOUN][ROOT=ev]
       [AGR=3SG][POSS=NONE][CASE=GEN]]

2.  ev+in       (your house)
     [[CAT=NOUN][ROOT=ev]
        [AGR=3SG][POSS=2SG][CASE=NOM]]
```

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. For instance the form `koyun` has the following parses:

```
1. koyu+[u]n  (your dark (thing))
    [[CAT=ADJ][ROOT=koyu]
      [CONV=NOUN=NONE]
      [AGR=3SG][POSS=2SG][CASE=NOM]]

2. koyun   (sheep)
    [[CAT=NOUN][ROOT=koyun]
      [AGR=3SG][POSS=NONE][CASE=NOM]]

3. koy+[n]un   (of the bay)
    [[CAT=NOUN][ROOT=koy]
      [AGR=3SG][POSS=NONE][CASE=GEN]]

4. koy+un   (your bay)
   [[CAT=NOUN][ROOT=koy]
      [AGR=3SG][POSS=2SG][CASE=NOM]]

5. koy+[y]un   (put !)
   [[CAT=VERB][ROOT=koy]
      [SENSE=POS][TAM1=IMP][AGR=2PL]]
```

4. The roots take unrelated inflectional and/or derivational suffixes which when concatenated turn out to have the same surface form. For instance the form `yapmadan` has the following parses:

```
1.  yap+madan   (without doing (it))
     [[CAT=VERB][ROOT=yap]
       [SENSE=POS][CONV=ADVERB=MADAN]]

2.  yap+ma+dan  (from doing (it))
    [[CAT=VERB][ROOT=yap]
      [SENSE=POS][CONV=NOUN=MA]
      [TYPE=INFINITIVE]
      [AGR=3SG][POSS=NONE][CASE=ABL]]
```

5. One of the ambiguous parses forms is a lexicalized form while one of the other is a form derived by a productive lexicalized form.

[1] Voutilainen, Private communication.

[2] Output of the morphological analyzer is edited for clarity, and English glosses have been given. We have also provided the morpheme structure, where [...]s, indicate elision.

[3] Glosses are given as linear feature value sequences corresponding to the morphemes (which are not shown). 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.

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

```
1. uygulama / application
   [[CAT=NOUN][ROOT=uygulama]
     [AGR=3SG][POSS=NONE][CASE=NOM]]

2.. uygula+ma /  ((the act of) applying)
    [[CAT=VERB][ROOT=uygula]
      [SENSE=POS][CONV=NOUN=MA]
      [TYPE=INFINITIVE]
      [AGR=3SG][POSS=NONE][CASE=NOM]]

3. uygula+ma / (don't apply!)
   [[CAT=VERB][ROOT=uygula]
     [SENSE=NEG][TAM1=IMP][AGR=2SG]]
```

It is certainly possible to remove or reduce some of the ambiguity by enforcing local linguistic constraints across a sequence of tokens, as is usually done in constraint-based disambiguation systems.

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):

$$Ambiguity = \frac{\#Parses}{\#Tokens}$$

$$Recall = \frac{\#Tokens\ Correctly\ Disambiguated}{\#Tokens}$$

$$Precision = \frac{\#Tokens\ Correctly\ Disambiguated}{\#Parses}$$

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,[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-based Morphological Disambiguation

This section outlines our approach to constraint-based morphological disambiguation where constraints vote on matching parses of sequential tokens.

[5]Assuming no unknown words.

3.1 Constraints on Morphological Parses

We describe constraints on the morphological parses of tokens using rules with two components

$$R = (C_1, C_2, \cdots, C_n, V)$$

where the C_i are (possibly hierarchical) feature constraints on a sequence of the morphological parses, and V is an integer denoting the weight of 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 which subcategorizes for an ablative nominal form.

   ```
   [[case:abl], [cat:postp,subcat:abl]]
   ```

2. The rule

   ```
   [[agr:'2SG',case:gen],[cat:noun,poss:'2SG']]
   ```

   chooses a nominal form with a possessive marker 2SG following a pronoun with 2SG agreement and genitive case, enforcing the simplest 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 following rule is an example of a rule employing a hierarchical constraint:

   ```
   [[cat:adj,stem:[tam1:narr]],
                     [cat:noun,stem:no]]
   ```

   which selects the derived participial adjectival reading of a verb with narrative past tense, if it is followed by an underived noun form.

3.2 Determining the vote of a rule

The vote of a rule is (in general) determined by its static properties. 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 $R = (C_1, C_2, \cdots, C_n, V)$ 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 a feature constraint $f_i : v_i$ 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 disambiguation. In this case, the weight of the feature constraint is $w(v_i)(> 1)$. For instance, our experience with Turkish indicates that a constraint referring to *genitive case* is very useful.
2. The feature itself may be a distinguished feature which has more important function in disambiguation. In this case the weight of the feature is $w(f_i)(> 1)$. For instance, the case of the nominal that a postposition subcategorizes for, (which is the *subcat* feature of the postposition) is crucial in disambiguation the item that precedes the postposition, hence is given a high value (see example 1 in Section 3.1.)
3. If the feature f_i refers to the stem of a derived form and the value part of the feature constraint 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 morphological constraint:

```
[cat:noun, case:gen,
 stem:[cat:adj, stem:[cat:v], suffix=mis]]
```

Assuming the value `gen` is a distinguished value with weight 4 (cf., rule 1 above), the vote of this constraint can be computed as follows:

- `cat:noun` contributes 1
- `case:gen` contributes 4
- `stem:[cat:adj, stem:[cat:v],suffix=mis]` contributes 8 computed as follows:
  - `cat:adj` contributes 1
  - `suffix=mis` contributes 1
  - `stem:[cat:v]` contributes $2 = 2 * 1$, the 1 being from `cat:v`.
  - the sum 4 is scaled by 2 to give 8.

So the resulting vote of the constraint is 13.

We also employ a set of rules which express preferences among the parses of single lexical form independent of the context in which the form occurs. The weights for these rules are currently manually determined. These rules give negative votes to the parses which are not preferred or high votes to certain parses which are always preferred. Our experience is that such preference rules depend on the kind of the text one is disambiguating. For instance if one is disambiguating a manual of some sort, imperative readings of verbs are certainly possible, whereas in normal plain text with no discourse, such readings are discouraged. Again in certain types of texts readings with `[POSS=2SG]` feature are very unlikely but yet they interfere with parses with `[AGR=3SG][POSS=NONE][CASE=GEN]` features. The most important use of these kinds of rules are in discouraging productively derived forms of certain words if there is already a lexicalized form (i.e., not involving the `STEM` feature). An example of a rule which penalizes such derived forms is:

```
[[stem:[stem:_]]]
```

which is associated with a large negative vote. This rule will penalize a parse with two productive derivations in it, and if there is a certain lexicalized parse for the same token[6], then the lexicalized parse will eventually survive.

3.3 Voting and Selecting Parses

A constraint $R = (C_1, C_2, \cdots, C_n, V)$ will match a sequence of tokens $w_i, w_{i+1}, \cdots, w_{i+n-1}$ within a sentence w_1 through w_s if *some morphological parse of every token* $w_j, i \leq j \leq i+n-1$ is subsumed by the corresponding constraint $\mathtt{C}_{j-i-1}$. When all constraints 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.[7]

After all rules have been applied to all token positions in a sentence and votes are tallied, morphological parses are selected in the following manner. Let v_l and v_h be the votes of the lowest and highest scoring parses for a given token. All parses with votes equal to or higher than $v_l + m * (v_h - v_l)$ are selected with m $(0 \leq m \leq 1)$ being a parameter. $m = 1$ selects the highest scoring parses.

4 Results from disambiguating Turkish text

We have applied our approach to disambiguating Turkish text. Raw text is processed by a *preprocessor* which segments the text into sentences using various heuristics about punctuation, and then tokenizes and runs it through a wide-coverage high-performance morphological analyzer developed using two-level morphology tools by Xerox (Karttunen, 1993). The preprocessor module also performs a number of additional functions such as grouping of *lexicalized* and *non-lexicalized* collocations, compound verbs, etc. The preprocessor also uses a second morphological processor for dealing with unknown words which recovers any derivational

[6]Most likely, the inflectional markers on both parses will be the same, but the lexicalized parse will have less derivations.

[7]An alternative strategy can be to divide the vote V among multiple matching parses of a given token.

and inflectional information from a word *even if the root word is not known.* This unknown word processor has a (nominal) root lexicon which recognizes S^+, where S is the Turkish surface alphabet (in the two-level morphology sense), but then tries to interpret an arbitrary postfix string of the unknown word, as a sequence of Turkish suffixes subject to all morphographemic constraints. Our experience with this unknown word processor is that, of all the occurrences of unknown words, over 96% eventually gets assigned the correct morphological parse (correct root, derivational markers and inflectional markers) after disambiguation.

We have applied our approach to four texts with statistics given in Table 1. The first text labeled ARK is a short text on near eastern archaeology. The second text labeled HIST is from a book on early 20^{th} century history of the Turkish Republic. The third text, MAN, is a computer manual text and the fourth text, EMB, is a foreign ministry text on embassy operations. In Table 1, the tokens considered are those that are generated after morphological analysis, unknown word processing and any lexical coalescing is done. The words that are counted as 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.[8] The fourth and fifth columns in this table give the average parses per token and the initial precision measure assuming initial recall is 1.0.

We have disambiguated these texts using a rule base of about 500 hand-crafted rules. Most of the rule crafting was done using the general linguistic constraints and constraints that we derived from the first text, ARK. In this sense, this text is our "training data", while the other three texts were not considered in rule crafting.

Our results are summarized in Table 2. The last four columns in this table present results for different values for the parameter m mentioned above, $m = 1$ denoting the case when only the highest scoring parse(s) is (are) selected.

The columns form $m < 1$ are given to emphasize that drastic change in precision for those cases. Even at $m = 0.95$ there is considerable loss of precision and going up to $m = 1.0$ 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 acceptable precision with just voting constraint rules. Our experience is that we can without any problem in principle add highly specialized rules by covering a larger text base to improve our recall and precision for the $m = 1.0$. The cases that have been missed are mostly due to morphosyntactic dependencies that span a context much wider that 5 tokens that we currently employ (i.e., the maximum number of constraints we have in our rules is 5.).

[8]The reason for the (comparatively) high number of unknown words in MAN, is that tokens found in such texts, like *f10*, denoting a function key in the computer can not be parsed as a Turkish root word!

		Vote Range Selected(m)				
TEXT		1.0	0.95	0.8	0.6	0.5
ARK	Rec.	98.05	98.47	98.69	98.77	98.80
	Prec.	94.13	87.65	84.41	82.43	81.98
	Amb.	1.042	1.123	1.169	1.200	1.205
HIST	Rec.	97.03	97.65	98.81	97.01	98.12
	Prec.	94.13	87.10	84.41	82.29	81.81
	Amb.	1.058	1.121	1.169	1.189	1.199
MAN	Rec.	97.03	97.92	97.81	98.77	98.77
	Prec.	91.05	83.51	79.85	77.34	76.69
	Amb.	1.068	1.172	1.237	1.277	1.288
EMB	Rec.	96.51	97.48	97.76	97.94	97.94
	Prec.	91.28	84.36	77.87	75.79	77.28
	Amb.	1.057	1.150	1.255	1.292	1.301

Table 2: Results with voting constraints

4.1 Using root and contextual statistics

We have employed two additional sources of information: *root word usage statistics, and contextual statistics.*

We have statistics compiled from previously disambiguated text on root frequencies. After the application of constraints as described above, for 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 lower than the frequency of the root of the highest scoring parse. The results after applying this step are shown in Table 3, where root statistics are applied to the voting results for $m = 1.0$ in Table 2.

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 in such an unambiguous context, 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.

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),

Text	Sent.	Tokens	Parses/ Token	Init. Prec.	Distribution of Morphological Parses					
					0	1	2	3	4	> 4
ARK	492	7928	1.823	0.55	0.15%	49.34%	30.93%	9.19%	8.46%	1.93%
HIST	270	5212	1.797	0.56	0.02%	50.63%	30.68%	8.62%	8.36%	1.69%
MAN	204	2756	1.840	0.54	0.65%	49.01%	31.70%	6.37%	8.91%	3.36%
EMB	198	5177	1.914	0.52	0.09%	43.94%	34.58%	9.60%	9.46%	2.33%

Table 1: Statistics on Texts

TEXT		m=1.0
ARK	Rec.	97.60
	Prec.	95.28
	Amb.	1.024
HIST	Rec.	96.52
	Prec.	92.59
	Amb.	1.042
MAN	Rec.	96.47
	Prec.	93.08
	Amb.	1.042
EMB	Rec.	96.47
	Prec.	93.08
	Amb.	1.036

Table 3: Results with voting constraints and root statistics

TEXT		m=1.0
ARK	Rec.	96.98
	Prec.	96.19
	Amb.	1.008
HIST	Rec.	95.62
	Prec.	94.33
	Amb.	1.013
MAN	Rec.	95.84
	Prec.	94.47
	Amb.	1.014
EMB	Rec.	95.37
	Prec.	94.45
	Amb.	1.009

Table 4: Results with voting constraints, root statistics, and contextual statistics

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

The current implementation of the voting approach is meant to be a proof of concept implementation and is rather inefficient. We are currently investigating an approach based on an adaptation of an efficient processing technique used in artificial intelligence for forward chaining rule-based production systems (e.g., (Forgy, 1981)), commonly used for implementing expert systems. This approach known as the RETE algorithm (Forgy, 1982) compiles a set of rules into an efficient discrimination network by combining most of the redundancies across rules by trading memory for repeated computation. Such systems maintain data items in what is known as a *working memory* and go through *match – execute* cycles. During the match phase, rules applicable in the current state of the working memory are found and then in the execute phase, the actions (such as delete, insert, modify) of the matching rules are executed on the contents of the working memory (Forgy, 1981).

In our case, the working memory corresponds to the set of parses of the tokens in a single sentence. We represent the parses of the tokens in a sentence as triples of position indices, parses, and votes:

```
(Position, Parse, Vote)
```

with `Position` indicating the linear position of the token in the sentence, `Parse` indicating (one of) the morphological parses, and `Vote` indicating the vote received (with an initial value of 0). The number of triples in the working memory during the processing of a sentence is equal to the total number of all the parses of all tokens in the sentence.

In this framework, we represent our rules that we described earlier as $R = (C_1, C_2, \cdots, C_n, V)$, as production system rules of the sort:

```
if (Position = X & $C_1$) &
   (Position = X+1, & $C_2$) &
   ...
   (Position = X+n-1, & $C_n$)
then
   increment-vote(1, V),
   increment-vote(2, V),
   ...
   increment-vote(n, V).
```

Such a rule is interpreted as follows: if there is a triple in the working memory with some position value (bound to variable X) matching the constraint

C_1, *and* there is a second triple whose position value is equal to $X+1$ (hence is for the next token) matching C_2, and so on, so that all n constraints are satisfied along with the sequencing constraints (imposed by the constraints on position values), *then* the rule is selected for "firing" and the **increment-vote** actions increment the count field of the respective matching triples. A sentence cycle is completed when all such rules for all relevant token positions in the sentence are found and their actions executed.

A RETE network is compiled from a rule base consisting of rules like the rule above and is a very compact representation. Rule conditions are checked through the discrimination network as working memory triples are inserted one by one, and when the last triple is inserted, all matching rules are found and the actions of matching rules are executed. We are currently working on a compiler that compiles our rules into such a discrimination network for efficient rule matching.

6 Conclusions

We have presented an approach to constraint-based morphological disambiguation which uses constraint voting as its primary mechanism for parse selection. Constraints describing language specific linguistic constraints or lexical preferences vote on matching parses of tokens, and at the end, parses for every token receiving the highest tokens are selected. Additional statistics are used afterwards to perform further disambiguation. The results at this stage are quite promising, and we expect to improve the recall and precision results as we add in further constraints. Our approach is quite general and is applicable to any language. Although the current way of assigning votes to rules is quite simple and justifiable, we feel that machine learning techniques can be applied to find the votes of features and values that may be more effective.

7 Acknowledgments

This research has been supported in part by a NATO Science for Stability Grant TU–LANGUAGE.

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 Twelfth National Conference on Articial Intelligence (AAAI-94)*, Seattle, Washinton.

Brill, Eric. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):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.

Forgy, Charles L. 1981. OPS5 user's manual. Technical report, Department of Computer Science, Carnegie Mellon University.

Forgy, Charles L. 1982. RETE: A fast algorithm for the many pattern/many object match problem. *Articial Intelligence*, 19(1).

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

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.

Voutilainen, Atro. 1994. *Three studies of grammar-based surface-syntactic parsing of unrestricted English text*. Ph.D. thesis, Research Unit for Computational Linguistics, University of Helsinki.

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.