Morphological Disambiguation of Turkish Text with Perceptron Algorithm

Haşim Sak[1], Tunga Güngör[1], and Murat Saraçlar[2]

[1] Dept. of Computer Engineering,
Boğaziçi University, Bebek, 34342, Istanbul, Turkey,
{hasim.sak,gungort}@boun.edu.tr,
[2] Dept. of Electrical and Electronic Engineering,
Boğaziçi University, Bebek, 34342, Istanbul, Turkey,
murat.saraclar@boun.edu.tr

Abstract. This paper describes the application of the perceptron algorithm to the morphological disambiguation of Turkish text. Turkish has a productive derivational morphology. Due to the ambiguity caused by complex morphology, a word may have multiple morphological parses, each with a different stem or sequence of morphemes. The methodology employed is based on ranking with perceptron algorithm which has been successful in some NLP tasks in English. We use a baseline statistical trigram-based model of a previous work to enumerate an n-best list of candidate morphological parse sequences for each sentence. We then apply the perceptron algorithm to rerank the n-best list using a set of 23 features. The perceptron trained to do morphological disambiguation improves the accuracy of the baseline model from 93.61% to 96.80%. When we train the perceptron as a POS tagger, the accuracy is 98.27%. Turkish morphological disambiguation and POS tagging results that we obtained is the best reported so far.

1 Introduction

Morphological disambiguation problem can be stated as finding the correct morphological parses of the words in a text given all the possible parses of the words. The parses can be obtained by using a morphological parser such as [1]. The morphological parsing of a word may result in multiple parses of that word due to the ambiguity in the root words and the morphemes, and the complex morphophonemic interaction between them ordered according to the morphotactics. Even to decide the part-of-speech tagging of a word, we may need to disambiguate the parses if they have different part-of-speech tags for the final derived word forms.

The agglutinative or inflective languages such as Turkish, Czech, Finnish, and Hungarian impose some difficulties in language processing due to the more complex morphology and relatively free word order in sentences when compared with languages like English. The morphemes carry syntactic and semantic information that is called morphosyntactic and morphosemantic features, respectively. Morphological disambiguation problem for these morphologically productive languages can also be considered as morphosyntactic tagging in analogy to part-of-speech tagging in other languages. The morphological disambiguation of text in these languages is required for further natural language processing tasks such as syntax parsing, word sense disambiguation, semantic parsing and analysis, language modeling for speech recogniton, etc. to be accomplished.

There have been generally two approaches to part-of-speech tagging. The rule-based approaches employ a set of hand-crafted linguistic rules that use the context information of a word to constrain the possible part-of-speech tags [2] or to assign a part-of-speech tag to that word [3]. These disambiguation rules can also be learned using transformation-based learning approach [4]. The statistical approaches select the most likely interpretation based on the estimation of statistics from unambiguously tagged text using a Markov model [5] or a maximum-entropy model [6] or ambiguously tagged text using a hidden Markov model [7].

The morphosyntactic tagging of agglutinative or inflective languages is more difficult due to the large number of tags. An exponential probabilistic model has been employed to tagging of the inflective language Czech [8]. Several constraint-based methods for morphological disambiguation in Turkish have been applied [9, 10]. A trigram-based statistical model has also been used in morphological disambiguation of Turkish text [11]. This model has also been used in this work as a baseline and will be discussed in later sections. A recent work has employed a decision list induction algorithm called Greedy Prepend Algorithm (GPA) to learn morphological disambiguation rules for Turkish [12].

The voted or averaged perceptron algorithms that have been previously applied to classification problems [13] have also been adapted very successfully to common NLP tasks such as syntax parsing of English text [14] and part-of-speech tagging and noun phrase chunking [15].

In this paper we describe the application of ranking with perceptron algorithm to morphological disambiguation of Turkish text. We use a baseline trigram-based model of a previous work to enumerate n-best candidates of morphological parse sequences of sentences. We then apply the perceptron algorithm to rerank the n-best list using a set of features. In the following sections, we first state the morphological disambiguation problem formally and describe the baseline model. We then present the perceptron algorithm and the features incorporated in the model. We conclude with the experiments and results.

2 Morphological Disambiguation

Turkish is an agglutinative language with a productive inflectional and derivational morphology. The complex morphology of Turkish allows thousands of word form to be constructed from a single root word using inflectional and derivational suffixes. The morphological parsing of a word may result in multiple interpretations of that word due to this complex morphology. Morphological disambiguation problem can be stated as finding the correct morphological parses of the words in a text given all the possible parses of the words.

The example below shows the multiple interpretations for the Turkish word `alın` with their parses as output from a Turkish morphological analyzer [1] and their English gloss.

```
alın+Noun+A3sg+Pnon+Nom (forehead)
al+Adj^DB+Noun+Zero+A3sg+P2sg+Nom (your red)
al+Adj^DB+Noun+Zero+A3sg+Pnon+Gen (of red)
al+Verb+Pos+Imp+A2pl ((you) take)
al+Verb^DB+Verb+Pass+Pos+Imp+A2sg ((you) be taken)
alın+Verb+Pos+Imp+A2sg ((you) be offended)
```

As can be seen, some of the parses have different root words and have unrelated morphological features due to the complex morphology of Turkish. These ambiguities mostly can be resolved using the contextual information, however the relatively free word order of Turkish also poses some difficulties in the sense that the limited context information cannot resolve the ambiguities. Some of the ambiguities can only be solved using semantic or discourse knowledge.

2.1 Representation

Agglutinative or inflective languages encode more information than just part-of-speech tag in a word thanks to the more complex morphology. The morphemes that constitute a word carry syntactic and semantic information that is called morphosyntactic and morphosemantic features, respectively. For morphological disambiguation, we need to determine all the syntactic morphological features of a word. Therefore morphological disambiguation can be called morphosyntactic tagging in analogy to part-of-speech tagging. We will use the same representation for the tags by Hakkani-Tür et al. in [11] where the full morphological parses of the words including the root words and their morphological features are treated as their morphosyntactic tags. An example that shows one of the morphological parses of the word `alın` consisting of the root word and some morphological features seperated using derivational boundary marker ˆDB is given below.

```
al+Adj^DB+Noun+Zero+A3sg+P2sg+Nom (your red)
```

Due to the productive inflectional and derivational morphology, the vocabulary size of Turkish can be very large. The large vocabulary size causes data sparseness problem and large number of out-of-vocabulary words when the word forms are considered as the units in a statistical model. This large vocabulary also prevents us from storing all the words and their possible tags in a lexicon. To alleviate the data sparseness problem and the inability of constructing a word form lexicon, they split the morphological parse of a word to its root and a sequence of inflectional groups (IGs) using derivational boundaries as shown below.

$root + IG_1\hat{}DB + IG_2\hat{}DB + ...\hat{}DB + IG_n$

In this way, instead of considering the morphological parse as a single unit, the inflectional groups can be treated as distinct units. As an example, the above

morphological parse can be written as a sequence of the root `al` and two inflectional groups.
`al+[Adj]+[Noun+Zero+A3sg+P2sg+Nom]`

2.2 Problem Definition

In this section, we formally define the morphological disambiguation problem using the representation of morphological parses described in the previous section. The problem can be stated as follows: given a sequence of words $W = w_1^n = w_1, w_2, \ldots, w_n$, find the corresponding sequence of morphological parses $T = t_1^n = t_1, t_2, \ldots, t_n$ of the words. Using the Bayesian approach, this can be formulated as follows:

$$\begin{aligned}\arg\max_T P(T|W) &= \arg\max_T \frac{P(T)P(W|T)}{P(W)} \\ &= \arg\max_T P(T)\end{aligned}$$

We can get rid of the $P(W)$ since it is constant for all morphological parses of the word and we can take $P(W|T)$ as equal to 1, since given the morphological parses we can uniquely determine the sequence of word forms assuming no morphological generation ambiguity. Therefore the problem has been reduced to finding the most probable parse sequence given all the possible parse sequences for a sentence.

2.3 Methodology

The problem of finding the most likely parse sequence given all the possible parse sequences for a sentence can be solved by estimating some statistics over the parts of the parses on a training set and choosing the most likely parse using the estimated parameters. This approach has been applied in trigram-based statistical model of Hakkani-Tür et al. in [11] using the root and inflectional groups as the units of the model to alleviate the data sparseness problem as described above. However this approach has not given competitive results for Turkish when compared to the POS tagging of English. The performance of their morphological disambiguation system is 93.95%. When their system is used as a POS tagger by considering the last POS tag assigned to the word in its parse, the performance is 96.07%.

Using their trigram-based model to assign probabilities to trigram parse sequences, we decoded an n-best list of candidate parses for a sentence using the Viterbi algorithm. Then we applied the perceptron algorithm to rank the candidates. The averaged or voted perceptron that we used for ranking has been applied successfully to a range of NLP tasks by Collins and Duffy in [14, 15]. We chose the perceptron method since it is very flexible in features that can be incorporated in the model and the parameter estimation method is very easy and just requires additive updates to a weight vector. This is also the first application of the perceptron algorithm to morphological disambiguation as far as we know. In the next sections we describe the baseline model and perceptron algorithm.

3 Baseline Trigram-based Model

Trigram-based probabilistic model of Hakkani-Tür et al. in [11] has been used as a baseline to enumerate n-best candidate parses with the Viterbi algorithm. Their method breaks up the morphosyntactic tags at each derivation boundary into groups of morphosyntactic features consisting of POS tag of the derived form and a sequence of inflectional features as described above. A simple trigram model is estimated from the statistics over the groups of morphosyntactic features (called inflectional groups).

Using a trigram tagging model and representing morphosyntatic tag t_i as a sequence of root form plus inflectional groups $(r_i, IG_{i,1}, \ldots, IG_{i,n_i})$, we can write $P(T)$ as follows:

$$\begin{aligned} P(T) &= \prod_{i=1}^{n} P(t_i | t_{i-2}, t_{i-1}) \\ &= \prod_{i=1}^{n} P((r_i, IG_{i,1}, \ldots, IG_{i,n_i}) | \\ &\qquad (r_{i-2}, IG_{i-2,1}, \ldots, IG_{i-2,n_{i-2}}), \\ &\qquad (r_{i-1}, IG_{i-1,1}, \ldots, IG_{i-1,n_{i-1}})) \end{aligned}$$

To estimate $P(T)$, they have made some assumptions: The first assumption is that a root word depends only on the roots of the previous two words. The second assumption is that the presence of IGs in a word only depends on the final IGs of the two previous words. These two assumptions lead to their first model which they report as giving the best results. This is the model that we used for the baseline model in this work.

Using these assumptions, $P(T)$ can be written as:

$$\begin{aligned} P(T) = \prod_{i=1}^{n} & (P(r_i | r_{i-2}, r_{i-1}) \\ & \prod_{k=1}^{n_i} P(IG_{i,k} | IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}})) \end{aligned}$$

We estimated the individual probabilities using the standard n-gram probability estimation methods from a morphologically disambiguated training set. Then we constructed a second order Markov model of the candidate morphological parses using the estimated morphosyntactic tag trigram probabilities for a sentence, and finally we used the Viterbi algorithm to decode the n-best candidates with their likelihoods.

4 Perceptron Algorithm

```
**Inputs:** Training examples $(x_i, y_i)$
**Initialization:** Set $\bar{\alpha} = 0$
**Algorithm:**
**For** $t = 1 \dots T, i = 1 \dots n$
    Calculate $z_i = \arg\max_{z \in \mathbf{GEN}(x_i)} \mathbf{\Phi}(x_i, z) \cdot \bar{\alpha}$
    **If** $(z_i \neq y_i)$ **then** $\bar{\alpha} = \bar{\alpha} + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$
**Output:** Parameters $\bar{\alpha}$
```

Fig. 1. A variant of the perceptron algorithm from Collins (see [15])

We have replicated the perceptron algorithm from Collins (see [15]) in Figure 1. This algorithm estimates the parameter vector $\bar{\alpha}$ using a set of training examples. The algorithm makes multiple passes (denoted by T) over the training examples. For each example, it finds the highest scoring candidate among all candidates using the current parameter values. If the highest scoring candidate is not the correct one, it updates the parameter vector $\bar{\alpha}$ by the difference of the feature vector representation of the correct candidate and the highest scoring candidate. This way of parameter update increases the parameter values for features in the correct candidate and downweights the parameter values for features in the competitor. The morphological disambiguation problem as formulated above can be used with this algorithm as follows:

– The training examples are the sentence $x_i = w^i_{[1:n_i]}$ and the morphological parse sequence $y_i = t^i_{[1:n_i]}$ pairs for $i = 1 \dots n$, where n is the number of training sentences and n_i is the length of the i'th sentence.
– The function $\mathbf{GEN}(x_i)$ maps the input sentence to the n-best candidate parse sequences using the baseline trigram-based model.
– The representation $\mathbf{\Phi}(x, y) \in \Re^d$ is a feature vector, the components of which are defined as $\Phi_s(w_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi_s(t_{i-2}, t_{i-1}, t_i)$, where $\phi_s(t_{i-2}, t_{i-1}, t_i)$ is an indicator function for a feature that depends on the current morphosyntactic tag (morphological parse) and the history of the previous two tags. Then the feature vector components $\Phi_s(w_{[1:n_i]}, t_{[1:n_i]})$ are just the counts of the local features $\phi_s(t_{i-2}, t_{i-1}, t_i)$. For example one feature might be:

$$\phi_{100}(t_{i-2}, t_{i-1}, t_i) = \begin{cases} 1 \text{ if current parse } t_i \\ \quad \text{is al+Verb+Pos} \\ \quad \text{+Imp+A2pl and} \\ \quad \text{previous parse } t_{i-1} \\ \quad \text{interpretation} \\ \quad \text{is a pronoun} \\ 0 \text{ otherwise} \end{cases}$$

– The expression $\mathbf{\Phi}(x, y) \cdot \bar{\alpha}$ in the algorithm is the inner product $\sum_s \alpha_s \Phi_s(x, y)$.

We used the "averaged parameters" to apply the method to the test examples since the averaged parameters are more robust to noisy or unseperable data [15]. The estimation of parameter values from training examples using the algorithm in Figure 1 is the same. The only difference is that we make a simple modification to the algorithm to sum the parameter values for each feature in a vector after each training example and the algorithm returns the averaged parameters γ by dividing this sum vector by the total number of examples used to update the vector. With this setting, the perceptron algorithm learns an averaged parameter vector γ that can be used to choose the most likely candidate morphological parse sequence of a sentence using the following function:

$$\begin{aligned} F(x) &= \arg \max_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x,y) \cdot \gamma \\ &= \arg \max_{y \in \mathbf{GEN}(x)} \gamma_0 \mathit{\Phi}_0(x,y) + \sum_{s=1}^{d} \mathit{\Phi}_s(x,y) \gamma_s \end{aligned}$$

where γ_0 is a weighting factor for the log probability $\mathit{\Phi}_0(x,y)$ assigned to the parse sequence by the baseline model. This parameter is found emprically as explained in the later sections.

Convergence theorems for the perceptron algorithm applied to tagging and parsing problems are given in [15].

5 Experiments

5.1 Data Set

We used a morphologically disambiguated Turkish corpus of about 950,000 tokens (including markers such as begin and end of sentence markers). Alternative ambiguous parses of the words are also available in the corpus as output from a morphological analyzer. This data set was divided into a training, development, and test set. The training set size is about 750,000 tokens or 45,000 sentences. The development set size is about 40,000 tokens or 2,500 sentences. The test set size is also about 40,000 tokens or 2,500 sentences. The training set was used to train the baseline trigram-based model and for the parameter estimation in perceptron algorithm. The development set was used to tune some of the parameters in the perceptron algorithm. The final tests were done on the test set.

5.2 Features

In the perceptron algorithm for morphological disambiguation we used a feature set that takes into account the current morphosyntactic tag (parse) and the history of the previous two tags. The set of features that we included in the model is shown in Table 1. In this table IG_i is the sequence of the inflection groups of the i'th morphosyntactic tag in the sentence. $IG_{i,j}$ is the j'th inflection group of the i'th morphosyntactic tag in the sentence. n_i is the number of inflection groups in the i'th morphosyntactic tag in the sentence.

Table 1. Features used for morphological disambiguation

Gloss	Feature
Trigram	(1) $r_{i-2}IG_{i-2}, r_{i-1}IG_{i-1}, r_iIG_i$
Bigram	(2) $r_{i-2}IG_{i-2}, r_iIG_i$
	(3) $r_{i-1}IG_{i-1}, r_iIG_i$
Current parse	(4) r_iIG_i
Previous parse and current IGs	(5) $r_{i-1}IG_{i-1}, IG_i$
Two previous parse and current IGs	(6) $r_{i-2}IG_{i-2}, IG_i$
Root trigram	(7) r_{i-2}, r_{i-1}, r_i
Root bigram	(8) r_{i-2}, r_i
	(9) r_{i-1}, r_i
Root unigram	(10) r_i
IGs Trigram	(11) IG_{i-2}, IG_{i-1}, IG_i
IGs Bigram	(12) IG_{i-2}, IG_i
	(13) IG_{i-1}, IG_i
IGs Unigram	(14) IG_i
for $j = 1 \ldots n_i$	(15) $IG_{i-2,n_{i-2}}, IG_{i-1,n_{i-1}}, IG_{i,j}$
n-grams using last IG of two previous	(16) $IG_{i-2,n_{i-2}}, IG_{i,j}$
parse and IG of current parse	(17) $IG_{i-1,n_{i-1}}, IG_{i,j}$
	(18) $IG_{i,j}$
for $j = 1 \ldots n_i - 1$ bigrams of IGs in current parse (local morphotactics)	(19) $IG_{i,j}IG_{i,j+1}$
for $j = 1 \ldots n_i$ IG and its position from the begining	(20) $j, IG_{i,j}$
Current parse is a proper noun and it starts with capital letter	(21) $PROPER$
Number of IGs in current parse	(22) $\#IG_i$
Current parse is a verb and it ends sentence	(23) $ENDSINVERB$

5.3 Optimal Parameter and Feature Selection

The free parameters in the perceptron algorithm are the number of iterations T and the weighting factor for the log probability $\Phi_0(x, y)$ assigned to the parse sequence by the baseline model. To optimize these parameters we ran the perceptron algorithm over the training set with varied parameters and tested on the development data to compare the results with different parameter values. We found that $T = 5$ iterations with $\gamma_0 = 0.0$ gives the best configuration for the parameters. The optimal weighting factor found to be 0.0 can be reasoned that the baseline model performance is comparatively very low and discarding the baseline log probability is better in this case.

We also did some experiments to select a subset of features that is optimal in terms of the accuracy of morphological disambiguation. The greedy algorithm that we used starts with no feature selected. Then it chooses the feature that improves the accuracy on the development set most. It continues in this manner with the remaining features until no feature increases the accuracy. Figure 2 shows the selected 9 features (4, 17, 3, 15, 20, 22, 9, 10, 2 - in this order) (see Table 1 for features referenced by these numbers) and the performance improvement when features are added.

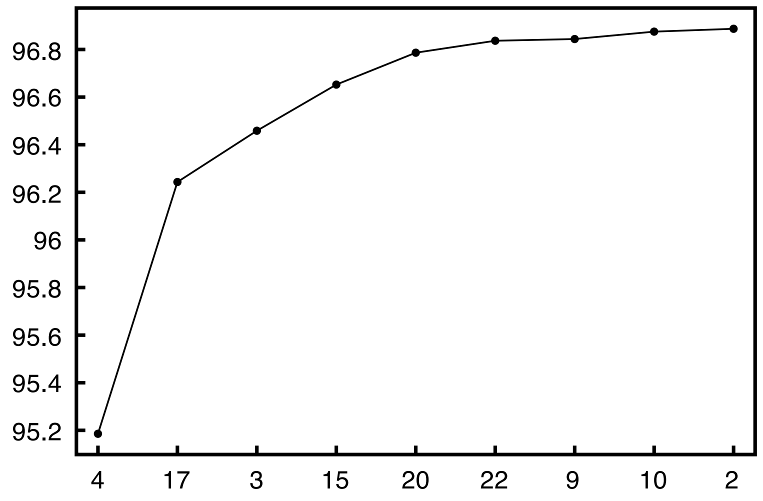

Fig. 2. Accuracy with respect to features added

5.4 Results

We first used the baseline trigram-based model to decode 50-best parses of each sentence in the data set. The training set was split to 5 portions and for each portion the baseline model was trained on the other 4 portions and that portion was decoded using the learned model. The development and test set was decoded using the baseline model trained with all the data in the training set. The baseline model also returns the log probability for each 50-best parses.

The baseline model performed with an accuracy of 93.61% on the test set. The perceptron algorithm was trained using the 50-best parse decodings of the training set. The parameter tuning was done using the 50-best parse decodings of the development set. The final test was done on the test set. Table 2 gives the accuracy results for the perceptron algorithm. The accuracy of the perceptron algorithm on the test set is 96.76% when all the 23 features are used and it is 96.80% when the 9 features (4, 17, 3, 15, 20, 22, 9, 10, 2) selected by the greedy method that we described above are used. The greedy method is effective in eliminating the non-discriminative features and hence increasing the runtime

Table 2. Ranking with Perceptron Results

Data set	Accuracy(%)
Perceptron (23 features)	96.76
Perceptron (9 features)	96.80

Table 3. Comparative Results on Test Set

Method	Error(%)
Baseline model	6.39
Perceptron (23 features)	3.24
Perceptron (9 features)	3.20

performance of the algorithm by reducing feature vector dimensions. For a comparision of the perceptron performance over the baseline model, see Table 3. The perceptron algorithm provides about 50% error reduction over the baseline model.

Error analysis for the morphological disambiguation experiment with 9 features shows that in 35% of errors (about 1.1% of all words) the root of the word is incorrectly decided. In 40% of errors the root is correct but its part of speech is incorrectly decided. In 17% of this case, the POS tag of the root is incorrectly decided as a noun in place of adjective. In 11%, noun should be pronoun, in 9% adjective should be noun, in 7% noun should be postposition, in 7% adjective should be determiner, in 5% noun should be adverb, in 5% adjective should be adverb and in 4% adverb should be adjective. In 25% of errors, root and its part of speech are correct but some inflection group is incorrect. In 16% of this case, a noun such as `kitabı` meaning in accusative case "the book" is incorrectly decided as a noun in nominative case meaning "his/her book". In 12%, the reverse is true. In 9%, the words that are derived from a verb using past participle suffix like `sevdiği` (beloved) is incorrectly labeled as adjective or noun.

We also ran the perceptron algorithm on a manually disambiguated small test set of 958 tokens to compare our results with Yüret and Türe in [12]. They have used the same train set in our experiments and tested on this small set. The comparative results can be seen in Table 4. The relatively inferior performance of the perceptron algorithm on this set can be explained by the small size of the test set and limited accuracy of the semi-automatically disambiguated train set.

The Turkish morphological disambigution performance using the perceptron algorithm (96.80%) is very close to the English part-of-speech tagging performance using the perceptron algorithm (97.11%) and maximum-entropy model (96.72%) as given in [15]. For a better comparison, when we consider the part-

Table 4. Comparative Results on Manually Tagged Test Set of 958 tokens

Method	Accuracy(%)
Baseline model	95.48
GPA (Yüret and Türe, 2006)	95.82
Perceptron (23 features)	96.28
Perceptron (9 features)	95.93

Table 5. Turkish POS tagging performance

POS tagger	Accuracy(%)
Baseline model	95.67%
Baseline model as reported in (Hakkani-Tür et al., 2002)	96.07%
MD perceptron (9 features)	98.19%
POS perceptron (9 features)	98.27%

of-speech tag of the word as given in the morphological parse of the word as the part-of-speech tag of the last derived form, the performance goes up to 98.19%. When we trained the perceptron to do POS tagging using the same 9 features used in the morphological disambiguation, the accuracy increased to 98.27%. The POS tagger performance for Turkish using perceptron algorithm is compared with the Turkish POS tagger performance as reported by Hakkani-Tür et al. in [11] in Table 5. We also presented our test result for the baseline model of Hakkani-Tür et al. on our test set in this table to make the comparison fair.

6 Conclusions

We presented an application of the perceptron algorithm to the morphological disambiguation of Turkish text. We used the Viterbi algorithm and a baseline trigram model to enumerate 50-best parses of a sentence. Then we ranked the candidates using the averaged perceptron algorithm. The perceptron algorithm provided about 50% error reduction over the baseline model. We found that a small set of features seems to be effective in morphological disambiguation of Turkish text. We also trained a perceptron for Turkish POS tagging which gives 98.27% accuracy. Turkish morphological disambiguation and POS tagging accuracy that we obtained is the best reported so far.

Acknowledgements

This work was supported by the Boğaziçi University Research Fund under the grant number 06A102 and partially supported by TUBITAK BIDEB 2211. The authors would like to thank Kemal Oflazer and Deniz Yüret for providing us the data set to conduct morphological disambiguation experiments.

References

1. Oflazer, K.: Two-level Description of Turkish Morphology. Literary and Linguistic Computing **9(2)** (1994) 137–148
2. Karlsson, F., Voutilainen, A., Heikkila, J., Anttila A.: Constraint Grammar-A Language-Independent System for Parsing Unrestricted Text (1995)
3. Brill, E.: A Simple Rule-Based Part-of-Speech Tagger. Proceedings of Third Conference on Applied Natural Language Processing, Trento, Italy (1992)
4. Brill, E.: Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging. Computational Linguistics (1995)
5. Church, K. W.: A stochastic parts program and noun phrase parser for unrestricted text. Proceedings of Second Conference on Applied Natural Language Processing, Austin, Texas (1988)
6. Ratnaparkhi, A.: A Maximum-Entropy Model for Part-of-Speech Tagging. Proceedings of the emprical methods in natural language processing conference (1996)
7. Cutting, D., Kupiec, J., Pealersen, J., Sibun, P.: A practical part-of-speech tagger. Proceedings of Third Conference on Applied Natural Language Processing, Trento, Italy (1992)
8. Hajič, J., Hladká, B.: Tagging inflective languages: prediction of morphological categories for a rich, structured tagset. Proceedings of COLING-ACL Conference (1998)
9. Oflazer, K., Tür, G.: Combining Hand-crafted Rules and Unsupervised Learning in Constraint-based Morphological Disambiguation. Proceedings of the ACL-SIGDAT Conference on Empirical Methods in Natural Language Processing, Philadelphia, PA, USA (1996)
10. Oflazer, K., Tür, G.: Morphological Disambiguation by Voting Constraints. Proceedings of ACL/EACL, The 35th Annual Meeting of the Association for Computational Linguistics, Madrid, Spain (1997)
11. Hakkani-Tür, D. Z., Oflazer, K., Tür, G.: Statistical Morphological Disambiguation for Agglutinative Languages. Computers and the Humanities **36(4)** (2002)
12. Yüret, D., Türe, F.: Learning Morphological Disambiguation Rules for Turkish. Proceedings of HLT-NAACL (2006)
13. Freund, Y., Schapire, R. E.: Large Margin Classification using the Perceptron Algorithm. Machine Learning **37(3)** (1999) 277–296
14. Collins, M., Duffy, N.: New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. Proceedings of ACL (2002)
15. Collins, M.: Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. Proceedings of EMNLP (2002)