Lang Resources & Evaluation (2011) 45:249–261
DOI 10.1007/s10579-010-9128-6

Resources for Turkish morphological processing

Haşim Sak · Tunga Güngör · Murat Saraçlar

Published online: 10 August 2010

Abstract We present a set of language resources and tools—a morphological parser, a morphological disambiguator, and a text corpus—for exploiting Turkish morphology in natural language processing applications. The morphological parser is a state-of-the-art finite-state transducer-based implementation of Turkish morphology. The disambiguator is based on the averaged perceptron algorithm and has the best accuracy reported for Turkish in the literature. The text corpus has been compiled from the web and contains about 500 million tokens. This is the largest Turkish web corpus published.

Keywords Turkish language resources · Morphological parser · Morphological disambiguation · Web corpus

1 Introduction

Turkish is an agglutinative language with a highly productive inflectional and derivational morphology which is quite regular (Lewis 2001; Göksel and Kerslake 2005). In morphologically rich languages, grammatical features and functions, which are associated with the syntactic structure of a sentence in other types of language, are often represented within the morphological structure of a word in

H. Sak (✉) · T. Güngör
Department of Computer Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey
e-mail: hasim.sak@boun.edu.tr

T. Güngör
e-mail: gungort@boun.edu.tr

M. Saraçlar
Department of Electrical & Electronic Engineering, Boğaziçi University, Bebek, 34342 Istanbul, Turkey
e-mail: murat.saraclar@boun.edu.tr

addition to the syntactic structure. In this respect, a number of linguistic theories such as Distributed Morphology (Halle 1993) and A-Morphous Morphology (Anderson 1992) relate morphology and syntax rather than considering morphology as concentrated in a single component of the grammar. Language applications for morphologically rich languages often require to exploit the syntactic and semantic information stored in the word structure. Therefore, we need some language resources and tools to extract and utilize this information.

In this paper, we describe the language resources that we built for processing the Turkish morphology. We make these language resources available for research purposes:

- A finite-state morphological parser (Sect. 2): It is a weighted lexical transducer that can be used for morphological analysis and generation of words. The transducer has been stochastized using the morphological disambiguator and the web corpus (Sak et al. 2009). The parser can be used with the OpenFST weighted finite-state transducer library (Allauzen et al. 2007).
- An averaged perceptron-based morphological disambiguator (Sect. 3): The proposed system has the highest disambiguation accuracy reported in the literature for Turkish. It also provides great flexibility in features that can be incorporated into the disambiguation model, parameter estimation is quite simple, and it runs very efficiently.
- A web corpus (Sect. 4): We aimed at collecting a representative sample of the Turkish language as it is used on the web. This corpus is the largest web corpus for Turkish.

2 Finite-state morphological parser

There are some previous computational studies on Turkish morphology. Oflazer (1994) gives a two-level morphological description implemented using the PC-KIMMO environment (Antworth 1990). However, its lexicon coverage is quite limited and it requires the PC-KIMMO system to run the parser which prevents the integration of the parser into other applications. Later, Oflazer has reimplemented this specification using Xerox finite-state tools,[1] *twolc* (a two-level rule compiler) (Karttunen and Beesley 1992) and *lexc* (a lexicon compiler). This implementation requires the Xerox software for execution and the parser is not publicly available. Öztaner (1996) also uses Xerox tools to build a morphological parser. Güngör (1995) describes Turkish morphophonemics and morphotactics using Augmented Transition Network formalism. While these studies consider only word-internal processes, Bozşahin (2002) proposes a morphemic grammar-lexicon for the integrated representation and processing of inflectional morphology, syntax, and semantics in a unified grammar architecture. Despite these studies, there is no publicly available state-of-the-art morphological parser for Turkish. Considering the success of finite-state machines in language and speech processing (Mohri 1997), it

[1] Personal communication.

is essential for a Turkish morphological parser to be available as a finite-state transducer in order to incorporate the morphology of the language as a knowledge source into other finite-state models.

In Turkish, theoretically one can produce an infinite number of words by inserting some derivational suffixes like the causative suffix in a word multiple times. Even if we ignore such iterations which are rarely used in practice, we can generate a word like the following using each suffix only once:

ölümsüzleştiriveremeyebileceklerimizdenmişsinizcesine

"(behaving) as if you are among those whom we could not cause hastily to become immortal"

We can break this word into morphemes as shown below:

$$\textit{ölüm} + \textit{süz} + \textit{leş} + \textit{tir} + \textit{iver} + \textit{eme} + \textit{yebil} + \textit{ecek} + \textit{ler} + \textit{imiz} + \textit{den} + \textit{miş} + \textit{siniz} + \textit{cesine}$$

In order to build a morphological parser, we need three components: a lexicon listing the stem words annotated with some information such as the part-of-speech tags to determine which morphological rules apply to them, a morphotactics component (morphosyntax) that describes the word formation rules by specifying the ordering of morphemes, and a morphophonemics component that describes the phonological alternations occurring in the morphemes during word formation. All these components can be implemented using finite-state transducers (FSTs).

We started with the specification of the Turkish morphology in the PC-KIMMO system (Oflazer 1994). The phonological rules and the morphotactics have been expanded and modified to cover the phenomena and the exceptions not handled in the PC-KIMMO implementation. We used the morphosyntactic tag set of Oflazer et al. (2003). Since the root word lexicon of the PC-KIMMO system with about 23,000 root words is limited in terms of word coverage and contains many misspelled words, we compiled a new lexicon of 55,278 root words based on the Turkish Language Institution dictionary (http://www.tdk.gov.tr/). We define "root word" as described by Aronoff (1993). The average number of parses per word on a morphologically disambiguated corpus of about 830,000 words parsed by Oflazer's parser is 1.86, while it is 2.30 when the same corpus is parsed by our parser.

The two-level rules (Koskenniemi 1984) that describe the phonological alternations in Turkish are compiled into a finite-state transducer (Kaplan and Kay 1994). For this purpose, we used the Xerox two-level rule compiler (Karttunen et al. 1987; Karttunen and Beesley 1992) . We composed the lexicon/morphotactics transducer with the morphophonemics transducer which is the intersection of the phonological rule transducers to build the lexical transducer of the parser (Karttunen et al. 1992). We used AT&T FSM tools (Mohri 1997) for finite-state operations. The resulting finite-state transducer can also be used with the OpenFST weighted finite-state transducer library (Allauzen et al. 2007).

We show below the morphological analysis of the word mentioned previously in this section as an example:

$$\begin{aligned} &\textit{ölüm}[\text{Noun}] - \textit{sHz}[\text{Adj} + \text{Without}] - \textit{lAş}[\text{Verb} + \text{Become}] - \textit{DHr}[\text{Verb} + \text{Caus}] \\ &\quad + [\text{Pos}] - \textit{YHver}[\text{Verb} + \text{Hastily}] + \textit{YAmA}[\text{Able} + \text{Neg}] - \textit{YAbil}[\text{Verb} + \text{Able}] \\ &\quad - \textit{YAcAk}[\text{Noun} + \text{FutPart}] + \textit{lAr}[\text{A3pl}] + \textit{HmHz}[\text{P1pl}] + \textit{NDAn}[\text{Abl}] \\ &\quad - \textit{YmHş}[\text{Verb} + \text{Narr}] + \textit{sHnHz}[\text{A2pl}] - \textit{CAsHnA}[\text{Adv} + \text{AsIf}] \end{aligned}$$

The morphological representation is similar to the one used in (Oflazer and Inkelas 2006). Each output of the parser begins with the root word and its part-of-speech tag in brackets. These are followed by a set of lexical morphemes associated with morphological features (nominal features such as case, person, and number agreement; verbal features such as tense, aspect, modality, and voice information). The inflectional morphemes start with a + sign. The derivational morphemes start with a − sign and the first feature of a derivational morpheme is the part-of-speech of the derived word form. A morphological feature may be appended without any morpheme, indicating that the feature is also applicable to the current word form.

The word coverage rate of the morphological parser is about 96.7% on the text corpus collected from online newspapers (see Table 3). The parser can also recognize the punctuation marks and the numerical tokens. It is highly efficient and can analyze about 8,700 words per second on a 2.33 GHz Intel Xeon processor. The morphological parser was also converted into a stochastic parser using the language resources described in this paper, which makes it the first stochastic morphological parser for Turkish (Sak et al. 2009).

3 Morphological disambiguation

The morphological parser may return more than one possible analysis for a word due to ambiguity. For example, the parser outputs four different analyses for the word *kedileri* as shown below. The English glosses are given in parentheses.

kedi[Noun]+*lAr*[A3pl]+*SH*[P3sg]+[Nom] (his/her cats)
kedi[Noun]+*lAr*[A3pl]+[Pnon]+*YH*[Acc] (the cats)
kedi[Noun]+*lAr*[A3pl]+*SH*[P3pl]+[Nom] (their cats)
kedi[Noun]+[A3sg]+*lArH*[P3pl]+[Nom] (their cat)

This parsing ambiguity needs to be resolved for further language processing using a morphological disambiguator (morphosyntactic tagger). There are several studies for morphosyntactic tagging in morphologically complex languages such as Czech (Hajic and Hladká 1998), which is an inflective language, and Basque (Ezeiza et al. 1998) and Hungarian (Megyesi 1999), which are agglutinative languages. For morphological disambiguation in Turkish, several constraint voting methods have been applied (Oflazer and Tür 1996, 1997). A statistical model has also been used (Hakkani-Tür et al. 2002), where statistics over inflectional groups

(chunks formed by splitting the morphological analysis of a word at derivation boundaries) are estimated by a trigram model. A recent work has employed a decision list induction algorithm called Greedy Prepend Algorithm (GPA) to learn morphological disambiguation rules for Turkish (Yüret and Türe 2006). The averaged perceptron algorithms previously applied to classification problems (Freund and Schapire 1999) have also been adapted very successfully to natural language processing (NLP) tasks such as syntactic parsing of English text (Collins and Duffy 2002) and part-of-speech tagging and noun phrase chunking (Collins 2002). This methodology was also proved to be quite successful for morphological disambiguation of Turkish text (Sak et al. 2007).

3.1 Methodology

The problem of finding the most likely morphological analyses of the words in a sentence can be solved by estimating some statistics over the parts of the morphological analyses on a training set and then choosing the most likely parse output using the estimated parameters. For parameter estimation, we use the averaged perceptron algorithm. This algorithm is very flexible in features that can be incorporated in the model and the parameter estimation just requires additive updates to a weight vector.

We presented an application of the averaged perceptron algorithm to morphological disambiguation of Turkish text in a previous study (Sak et al. 2007). In that study, a baseline trigram-based model of Hakkani-Tür et al. (2002) is used to enumerate n-best candidates of alternative morphological parses of a sentence. Then the averaged perceptron algorithm is applied to rerank the n-best candidate list using a set of features. In the present study, we do not use a baseline model to generate n-best candidates. Instead, we do a Viterbi decoding (Viterbi 1967) of the best path in the network of ambiguous morphological parses of the words in a sentence.

We split the morphological analysis of a word into morphemic units to be used as features by the perceptron algorithm. For this purpose we make use of the morpheme boundaries (both inflectional and derivational ones) in the analysis. This representation is different than the one used by Hakkani-Tür et al. (2002) and Sak et al. (2007), where only derivational boundaries are used to split the morphological analysis of a word into chunks called inflectional groups. A morphosyntactic tag t_i, which is a morphological analysis of a word w_i, is split into a root tag r_i and a morpheme tag m_i. The morpheme tag m_i is the concatenation of the morphosyntactic tags of morphemes $m_{i,j}$ for $j = 1, \ldots, n_i$, where n_i is the number of morphemes in t_i:

$$t_i = r_i m_i = r_i m_{i,1} m_{i,2}, \ldots, m_{i,n_i}$$

For example, the morphological analysis of the word $w_i = ulaşmadıgı$

$$t_i = ulaş[\text{Verb}] + mA[\text{Neg}] - DHk[\text{Noun+PastPart}] + [\text{A3sg}] + SH[\text{P3sg}] + [\text{Nom}]$$

is represented as its root tag and morpheme tags as follows:

Table 1 Feature templates used for morphological disambiguation

Gloss	Feature
Morphological parse trigram	(1) $t_{i-2}\ t_{i-1}\ t_i$
Morphological parse bigram	(2) $t_{i-2}\ t_i$ & (3) $t_{i-1}\ t_i$
Morphological parse unigram	(4) t_i
Morpheme tag with previous tag	(5) $t_{i-1}\ m_i$
Morpheme tag with second to previous tag	(6) $t_{i-2}\ m_i$
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
Morpheme tag trigram	(11) $m_{i-2}\ m_{i-1}\ m_i$
Morpheme tag bigram	(12) $m_{i-2}\ m_i$ & (13) $m_{i-1}\ m_i$
Morpheme tag unigram	(14) m_i
Individual morpheme tags	(15) $m_{i,j}$ for $j = 1, \ldots, n_i$
Individual morpheme tags with position	(16) $j\ m_{i,j}$ for $j = 1, \ldots, n_i$
Number of morpheme tags	(17) n_i

$$
\begin{aligned}
r_i &= ulaş[\text{Verb}] \\
m_{i,1} &= +mA[\text{Neg}] \\
m_{i,2} &= -DHk[Noun + PastPart] + [A3sg] \\
m_{i,3} &= +SH[P3sg] + [Nom]
\end{aligned}
$$

The set of features that we incorporate in the model is a subset of the features used by Sak et al. (2007). The feature set takes into account the current morphosyntactic tag t_i, the previous tag t_{i-1}, and the second to previous tag t_{i-2}. The feature templates are given in Table 1. We basically add unigram, bigram and trigram features over the root and the morpheme tags. The discriminative training algorithm of the perceptron learns the feature weights for each instance of these features.

3.2 Perceptron Algorithm

A variant of the perceptron algorithm is repeated in Fig. 1 from Collins (2002). The algorithm estimates a parameter vector α using a set of training examples (x_i, y_i),

Inputs: Training examples (x_i, y_i)
Initialization: Set $\alpha = 0, \gamma = 0$
Algorithm:
For $t = 1 \ldots T, i = 1 \ldots n$
  Calculate $z_i = \text{argmax}_{z \in \mathbf{GEN}(x_i)}\ \mathbf{\Phi}(x_i, z) \cdot \alpha$
  If $(z_i \neq y_i)$ **then** $\alpha = \alpha + \mathbf{\Phi}(x_i, y_i) - \mathbf{\Phi}(x_i, z_i)$
  $\gamma = \gamma + \alpha$
Output: Parameters $\gamma = \gamma/(nT)$

Fig. 1 A variant of the perceptron algorithm by Collins (2002)

which will be used for mapping from inputs $x \in X$ to outputs $y \in Y$. In our setting, X is a set of sentences and Y is a set of possible morphological parse sequences. 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 parse, it updates the parameter vector α 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. For the application of the model to the test examples, the algorithm calculates the "averaged parameters" since they are more robust to noisy or inseparable data (Collins 2002). The averaged parameters γ are calculated by summing the parameter values for each feature after each training example and dividing this sum by the total number of examples. The perceptron algorithm is adapted to the disambiguation problem as follows:

- The training examples are the pairs (x_i, y_i) for $i = 1, \ldots, n$, where n is the number of training sentences. For the ith sentence, x_i is the word sequence $w^i_{[1:n_i]}$ and y_i is the correct morphosyntactic tag sequence $t^i_{[1:n_i]}$, where n_i is the number of words in the sentence.
- The function $\mathbf{GEN}(x_i)$ maps the input sentence x_i to the candidate parse sequences. In the actual implementation, since the features depend on the current and previous two tags, we generate a network of parse outputs on-the-fly and do a Viterbi decoding of the best path without enumerating all the paths.
- The representation $\mathbf{\Phi}(x, y) \in \Re^d$ is a d-dimensional feature vector. Each component $\Phi_j(w_{[1:n]}, t_{[1:n]})$ for $j = 1, \ldots, d$ is the count of a local feature (n is the number of words in the word sequence x) and is defined as $\sum_{i=1}^{n} \rho_j(t_{i-2}, t_{i-1}, t_i)$, where $\rho_j(t_{i-2}, t_{i-1}, t_i)$ is an indicator function for the jth feature. The features depend on the current morphosyntactic tag (morphological parse) and the history of the previous two tags. An example indicator function for a feature (corresponding to feature template (9) in Table 1) might be:

$$\rho_{100}(t_{i-2}, t_{i-1}, t_i) = \begin{cases} 1 & \text{if the root tag of } t_i \text{ is gör[Verb] and} \\ & \text{the root tag of } t_{i-1} \text{ is uygun[Adj]} \\ 0 & \text{otherwise} \end{cases}$$

For this example, $\Phi_{100}(w_{[1:n]}, t_{[1:n]})$ is the number of times the root tag *uygun*[Adj] is followed by the root tag *gör*[Verb] in the tag sequence $t_{[1:n]}$. Then, the algorithm will update the 100th component of the parameter vector α by adding $\Phi_{100}(x, y) - \Phi_{100}(x, z)$, where y is the correct tag sequence and z is the highest scoring candidate.

- The expression $\mathbf{\Phi}(x, y) \cdot \alpha$ denotes the inner product $\sum_{j=1}^{d} \Phi_j(x, y)\alpha_j$, where α_j is the jth component of the parameter vector α.
- The function $\text{argmax}_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \alpha$ can be efficiently calculated using dynamic programming since the features that we use depend on only the current tag and the previous two tags.

With this setting, the perceptron algorithm learns an averaged parameter vector γ that is used to choose the most likely morphological parse sequence of a test sentence x using the following function:

$$F(x) = \text{argmax}_{y \in \mathbf{GEN}(x)} \mathbf{\Phi}(x, y) \cdot \gamma = \text{argmax}_{y \in \mathbf{GEN}(x)} \sum_{j=1}^{d} \Phi_j(x, y)\gamma_j$$

3.3 Experiments and results

In order to measure the performance of the morphological disambiguator, we used a semi-automatically disambiguated Turkish corpus of about 950,000 tokens (including markers such as begin and end of sentence markers) that has been tagged with Oflazer's parser. In addition to the correct morphological analysis of a word, alternative ambiguous analyses are also available in the corpus given as output by the morphological analyzer. The output format of the morphological parser developed in this work is somewhat different from that used in the corpus due to the improvements on the phonological rules and the morphotactics. Therefore, we converted the corpus to the parser format used in this work in order to be able to train a disambiguation model over the outputs of our morphological parser. In this conversion, the morphological analysis of 95.5% of the tokens were mapped automatically to an analysis in the form used by the morphological parser. The 2.6% of the tokens (mostly misspelled words and proper nouns) could not be parsed and they were marked as unknown words. The remaining 1.9% of the tokens were mapped to an analysis which has the minimum edit distance with the original parse.

This data set was divided into training, development and test sets with respective sizes of about 850,000 (45,000), 47,000 (2,500) and 48,000 (2,500) tokens (sentences). The training set was used for parameter estimation and the development set was used for feature selection. The trained model was tested on this semi-automatically disambiguated test set. We also tested the same trained model on a manually disambiguated test set (converted into our parser format) of 958 tokens. The accuracies on these two test sets were obtained as 97.05% (the 95% confidence interval is [0.9689, 0.9720]) and 96.45% (the 95% confidence interval is [0.9526, 0.9763]), respectively.

We also compared the performance of the perceptron algorithm on the original manually disambiguated test corpus of 958 tokens with previous studies on this corpus. The results are given in Table 2. The trigram-based model of Hakkani-Tür et al. (2002) is our implementation. The confidence interval for the 96.45% accuracy indicates that the performance improvement of the perceptron model over the other models is not statistically significant and we need a larger manually

Table 2 Comparative results on manually tagged test set (958 tokens)

Method	Accuracy (%)
Trigram-based model (Hakkani-Tür et al. 2002)	95.92
GPA (Yüret and Türe 2006)	95.82
Perceptron (this study)	96.45

disambiguated corpus to verify the performance improvements. We also measured the accuracy of the perceptron algorithm on the ambiguous cases only. In the test set containing 958 tokens, 579 tokens are unambiguous (96 of them are markers). The disambiguation accuracy on the 379 ambiguous tokens is 91.0%, while the baseline accuracy (selecting a parse randomly) is 38.5%.

The morphological disambiguator is also a part-of-speech (POS) tagger when we consider the POS tag of a word as the POS tag of the last derived word form as given in the morphological parse of the word. The POS tagging performance of the disambiguator is about 98.6% for both test sets.

The perceptron algorithm trains a disambiguation model by making four passes over the training examples of about one million tokens in one hour on a 2.33 GHz Intel Xeon processor. The generated model contains about 580,000 features. This is the dimension of the feature vector. It can disambiguate 1,000 words per second on the same processor.

4 Web corpus

Due to the productive morphology and parsing ambiguity in agglutinative languages, we need a large corpus of sentences for robust parameter estimation in statistical NLP models. There have been very few efforts to build a Turkish text corpus. METU Turkish Corpus is a hand-compiled annotated collection of two million words of written Turkish samples (Say et al. 2002). Another effort is the collection of web pages of Turkish newspapers containing about 2.5 million words for Turkish speech recognition research (Salor et al. 2002). These corpora are limited in size and coverage to be successfully used in many statistical natural language applications.

It has been reported that probabilistic models of language based on a very large corpus, even if the corpus is noisy, are better than those based on estimates from smaller, cleaner data sets (Kilgarriff and Grefenstette 2003). The web is being increasingly used by researchers to build web corpora (Liu and Curran 2006), since the other language resources are not large enough or not suitable in terms of language coverage and the web is instantly and freely available.

In this research, we built a web corpus for Turkish and cleaned it using the morphological parser and some heuristics. The Turkish web corpus (*BOUN Corpus*) is composed of four subcorpora. Three of these (Milliyet, Ntvmsnbc, Radikal) are from three major news portals in Turkish (collectively referred as *NewsCor*) and the other one (*GenCor*) is a general sampling of Turkish web pages. The statistics about the numbers of words, tokens (words and lexical units such as punctuation marks), and types (distinct tokens) are shown in Table 3. The percentages of the tokens and types that can be parsed by the morphological parser are also indicated. We observe that, due to the agglutinative nature of the language, the number of types (4.1M) is quite large. Also, the number of types parsed (1.57M) in the corpus being about 30 times larger than the size of the root lexicon of the parser indicates that derived words are used commonly in Turkish. For the encoding of the web corpus, we used

Table 3 Web corpus size and results of morphological parser

Corpus	Words	Tokens	Types	Tokens parsed (%)	Types parsed (%)
Milliyet	59M	68M	1.1M	96.7	63.5
Ntvmsnbc	75M	86M	1.2M	96.4	55.8
Radikal	50M	58M	1.0M	97.0	65.7
NewsCor	184M	212M	2.2M	96.7	52.2
GenCor	239M	279M	3.0M	94.6	39.5
BOUN corpus	423M	491M	4.1M	95.5	38.4

the XML Corpus Encoding Standard, XCES (http://xces.org) as used by Say et al. (2002).

4.1 Corpus statistics

In this section, we present some corpus statistics in order to get an idea about the coverage of the corpus and also to observe the morphological characteristics of the Turkish language. Figure 2 shows statistics about the types relative to the corpus size (number of tokens) excluding the numerical tokens. As can be seen, the number of types is increasing continuously for both corpora and for the combined corpus. For instance, when the corpus size was increased from 490M to 491M, 5,539 new types (of which 1,009 can be parsed successfully) have been added to the corpus. This situation is mostly due to misspelled word forms and partly due to the productive morphological structure of Turkish and the rich web environment. The analysis of a small sample of these 4,530 types that cannot be parsed shows that

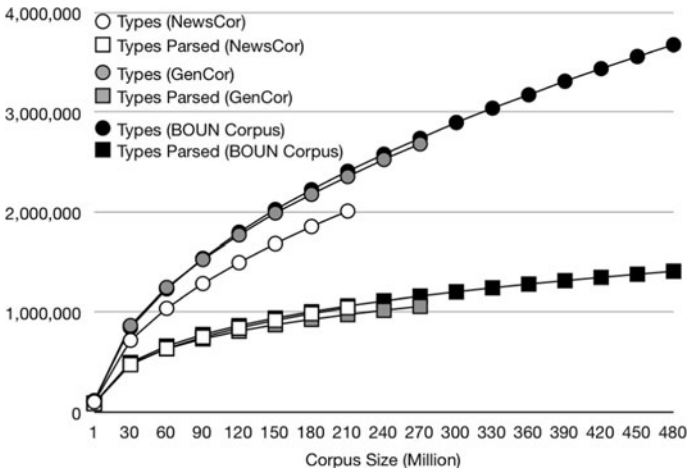

Fig. 2 Type statistics for subcorpora and combined corpus

about 70% of them are spelling errors, 20% are foreign words, and 10% are proper names.

We also analyzed the corpus coverage statistics with respect to the vocabulary size (number of types). The most frequent 50K types cover only 89% of the corpus, while 300K types are necessary in order to attain a coverage ratio of about 97%. The agglutinative nature of the language and the diversity of the web contents are the basic reasons of this result. A similar statistic related to the percentages of infrequent types shows that almost half of the types (about 2.0M) occur only once in the corpus. The number of types occurring less than 10 times is 3.4M. Thus, we see that the majority of the types in the corpus are very infrequent.

To understand the source of the large number of types in the corpus, we give statistics for the stems and lexical endings (tokens stripped of their stems in lexical form such as *+lAr+Hn*) in Fig. 3. As the corpus size increases, the number of unique stems approaches to the size of our lexicon (55,278 root words). The number of unique endings increases steadily as new data are added. This increase means that people freely derive new word forms by making use of suffix combinations not used before. Although we know that theoretically there is no limit on the number of derivations in Turkish, we might expect that in practice a (large) subset of all possible derived forms will cover the daily use of the language. However, this expectation does not hold even for a corpus of nearly 500M tokens; in the last 1M tokens of the corpus 5 new stems and 32 new lexical endings emerged and for the last 10M tokens these numbers were 61 and 268, respectively. The number of unique stems in the corpus is about 49,000, which is significantly less than our root lexicon size. Therefore, we can conclude that we need some more text from other domains to see all the root words. On the other hand, we expect that the statistics for lexical endings are not domain dependent. All the analyses have been done after randomly ordering the documents in the corpus.

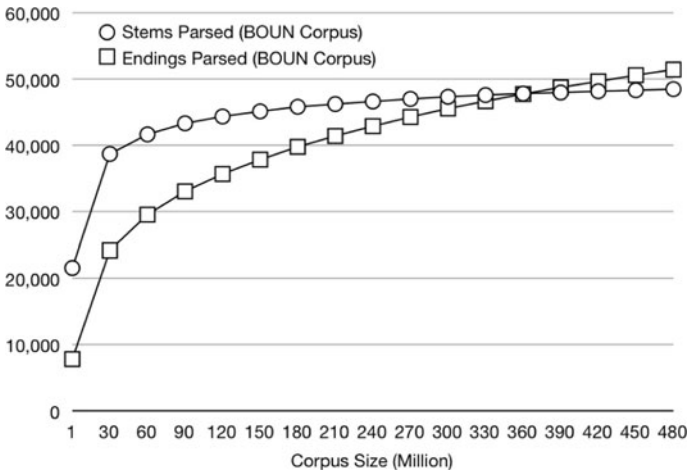

Fig. 3 Stem and lexical ending statistics for combined corpus

5 Conclusions

In this paper, we presented some essential tools and resources for exploiting the Turkish morphology in natural language processing applications. Morphology is a very important knowledge source for morphologically complex languages like Turkish. Using these resources and tools, one can parse a text corpus and obtain the morphological analyses of the words as well as their probabilities, disambiguate the parse outputs, train statistical models using the web corpus, and build applications that fully exploit the information hidden in the morphological structure of words. These resources are available for non-commercial research purposes.[2]

Acknowledgments This work was supported by the Boğaziçi University Research Fund under the grant numbers 06A102 and 08M103, the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the grant number 107E261, the Turkish State Planning Organization (DPT) under the TAM Project number 2007K120610. Murat Saraçlar is supported by the TUBA-GEBIP award. Haşim Sak is supported by TÜBİTAK BİDEB 2211. The authors would like to thank to Kemal Oflazer and Deniz Yüret for the disambiguation data set.

References

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., & Mohri, M. (2007). OpenFst: A general and efficient weighted finite-state transducer library. In *CIAA*, pp. 11–23.

Anderson, S. (1992). *A-Morphous morphology*. Cambridge: Cambridge University Press.

Antworth, E. L. (1990). PC-KIMMO: A two-level processor for morphological analysis. In *Occasional Publications in Academic Computing*.

Aronoff, M. (1993). *Morphology by itself: Stems and inflectional classes*. Cambridge: MIT Press.

Bozşahin, C. (2002). The combinatory morphemic lexicon. *Computational Linguistics, 28*(2), 145–186.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP*.

Collins, M., & Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *ACL*, pp. 263–270.

Ezeiza, N., Alegria, I., Arriola, J. M., Urizar, R., & Aduriz, I. (1998). Combining stochastic and rule-based methods for disambiguation in agglutinative languages. In *COLING-ACL*.

Freund, Y., & Schapire, R. (1999). Large margin classification using the perceptron algorithm. *Machine Learning, 37*(3), 277–296.

Göksel, A., & Kerslake, C. (2005). *Turkish: A comprehensive grammar*. London: Routledge.

Güngör, T. (1995). Computer processing of Turkish: Morphological and lexical investigation. Ph.D. thesis, Boğaziçi University.

Hajic, J., & Hladká, B. (1998). Tagging inflective languages: Prediction of morphological categories for a rich, structured tagset. In *COLING-ACL*, pp. 483–490.

Hakkani-Tür, D. Z., Oflazer, K., & Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities, 36*(4).

Halle, M., & Marantz, A. (1993). Distributed morphology and the pieces of inflection. In *The View from Building 20* (pp 111–176). Cambridge: MIT Press.

Kaplan, R. M., & Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics, 20*(3), 331–378.

Karttunen, L., & Beesley, K. R. (1992). Two-level rule compiler, Technical report. Palo Alto, CA: Xerox Palo Alto Research Center.

Karttunen, L., Koskenniemi, K., & Kaplan, R. M. (1987). A compiler for two-level phonological rules. In *Tools for morphological analysis*. Palo Alto, CA: Center for the Study of Language and Information, Stanford University.

[2] All resources are available at http://www.cmpe.boun.edu.tr/∼hasim.

Karttunen, L., Kaplan, R. M., & Zaenen, A. (1992). Two-level morphology with composition. In *COLING*, 141–148.
Kilgarriff, A., & Grefenstette, G. (2003). Introduction to the special issue on the web as corpus. *Computational Linguistics, 29*(3), 333–348.
Koskenniemi, K. (1984). A general computational model for word-form recognition and production. In *ACL*, pp. 178–181.
Lewis, G. (2001). *Turkish grammar*. Oxford: Oxford University Press.
Liu, V., & Curran, J. R. (2006). Web text corpus for natural language processing. In *EACL*.
Megyesi, B. (1999). Improving Brill's PoS tagger for an agglutinative language. In *EMNLP/VLC*.
Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics, 23*(2), 269–311.
Oflazer, K. (1994). Two-level description of Turkish morphology. *Literary and Linguistic Computing, 9*(2), 137–148.
Oflazer, K., & Inkelas, S. (2006). The architecture and the implementation of a finite state pronunciation lexicon for Turkish. *Computer Speech and Language, 20*(1), 80–106.
Oflazer, K., & Tür, G. (1996). Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. In *EMNLP*, (pp. 69–81). Somerset, NJ: ACL.
Oflazer, K., & Tür, G. (1997). Morphological disambiguation by voting constraints. In *ACL*, (pp. 222–229).
Oflazer, K., Say, B., Hakkani-Tür, D. Z., & Tür, G. (2003). Building a Turkish treebank. In *Building and exploiting syntactically-annotated corpora*. Dordrecht: Kluwer.
Öztaner, S. M. (1996). A word grammar of Turkish with morphophonemic rules. Master's thesis, Middle East Technical University.
Sak, H., Güngör, T., & Saraçlar, M. (2007). Morphological disambiguation of Turkish text with perceptron algorithm. In *CICLing 2007*, (vol. LNCS 4394, pp. 107–118).
Sak, H., Güngör, T., & Saraçlar, M. (2009). A stochastic finite-state morphological parser for Turkish. In *ACL-IJCNLP 2009*, (pp. 273–276).
Salor, Ö., Pellom, B. L., Ciloglu, T., Hacioglu, K., & Demirekler, M. (2002). On developing new text and audio corpora and speech recognition tools for the Turkish language. In *ICSLP*.
Say, B., Zeyrek, D., Oflazer, K., & Özge, U. (2002). Development of a corpus and a treebank for present-day written Turkish. In *Proceedings of the eleventh international conference of Turkish linguistics*.
Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory, 13*(2), 260–269.
Yüret, D., & Türe, F. (2006). Learning morphological disambiguation rules for Turkish. In *HLT-NAACL*.