A Stochastic Finite-State Morphological Parser for Turkish

Haşim Sak & Tunga Güngör
Dept. of Computer Engineering
Boğaziçi University
TR-34342, Bebek, İstanbul, Turkey
hasim.sak@boun.edu.tr
gungort@boun.edu.tr

Murat Saraçlar
Dept. of Electrical & Electronics Engineering
Boğaziçi University
TR-34342, Bebek, İstanbul, Turkey
murat.saraclar@boun.edu.tr

Abstract

This paper presents the first stochastic finite-state morphological parser for Turkish. The non-probabilistic parser is a standard finite-state transducer implementation of two-level morphology formalism. A disambiguated text corpus of 200 million words is used to stochastize the morphotactics transducer, then it is composed with the morphophonemics transducer to get a stochastic morphological parser. We present two applications to evaluate the effectiveness of the stochastic parser; spelling correction and morphology-based language modeling for speech recognition.

1 Introduction

Turkish is an agglutinative language with a highly productive inflectional and derivational morphology. The computational aspects of Turkish morphology have been well studied and several morphological parsers have been built (Oflazer, 1994), (Güngör, 1995).

In language processing applications, we may need to estimate a probability distribution over all word forms. For example, we need probability estimates for unigrams to rank misspelling suggestions for spelling correction. None of the previous studies for Turkish have addressed this problem. For morphologically complex languages, estimating a probability distribution over a static vocabulary is not very desirable due to high out-of-vocabulary rates. It would be very convenient for a morphological parser as a word generator/analyzer to also output a probability estimate for a word generated/analyzed. In this work, we build such a stochastic morphological parser for Turkish[1] and give two example applications for evaluation.

[1]The stochastic morphological parser is available for research purposes at http://www.cmpe.boun.edu.tr/~hasim

2 Language Resources

We built a morphological parser using the two-level morphology formalism of Koskenniemi (1984). The two-level phonological rules and the morphotactics were adapted from the PC-KIMMO implementation of Oflazer (1994). The rules were compiled using the *twolc* rule compiler (Karttunen and Beesley, 1992). A new root lexicon of 55,278 words based on the Turkish Language Institution dictionary[2] was compiled. For finite-state operations and for running the parser, we used the OpenFST weighted finite-state transducer library (Allauzen et al., 2007). The parser can analyze about 8700 words per second on a 2.33 GHz Intel Xeon processor.

We need a text corpus for estimating the parameters of a statistical model of morphology. For this purpose, we compiled a text corpus of 200 million-words by collecting texts from online newspapers. The morphological parser can analyze about 96.7% of the tokens.

The morphological parser may output more than one possible analysis for a word due to ambiguity. For example, the parser returns four analyses for the word *kedileri* as shown below. The morphological representation is similar to the one used by Oflazer and Inkelas (2006).

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)

We need to resolve this ambiguity to train a probabilistic morphology model. For this purpose, we used our averaged perceptron-based morphological disambiguator (Sak et al., 2008). The disambiguation system achieves about 97.05% disambiguation accuracy on the test set.

[2]http://www.tdk.gov.tr

Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 273–276,
Suntec, Singapore, 4 August 2009.

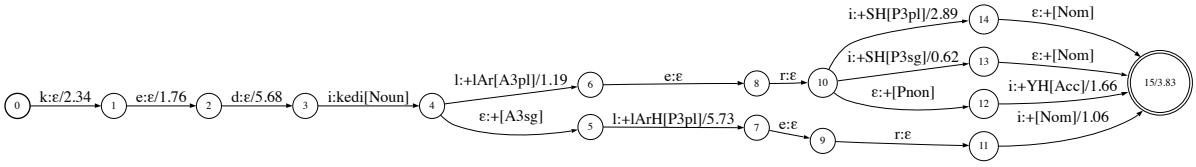

Figure 1: Finite-state transducer for the word *kedileri*.

3 Stochastic Morphological Parser

The finite-state transducer of the morphological parser is obtained as the composition of the morphophonemics transducer mp and the morphotactics transducer mt; $mp \circ mt$. The morphotactics transducer encodes the morphosyntax of the language. If we can estimate a statistical morphosyntactic model, we can convert the morphological parser to a probabilistic one by composing the probabilistic morphotactics transducer with the morphophonemics transducer. Eisner (2002) gives a general EM algorithm for parameter estimation in probabilistic finite-state transducers. The algorithm uses a bookkeeping trick (expectation semiring) to compute the expected number of traversals of each arc in the E step. The M step reestimates the probabilities of the arcs from each state to be proportional to the expected number of traversals of each arc - the arc probabilities are normalized at each state to make the finite-state transducer Markovian. However, we do not need this general method of training. Since we can disambiguate the possible morphosyntactic tag sequences of a word, there is a single path in the morphotactics transducer that matches the chosen morphosyntactic tag sequence. Then the maximum-likelihood estimates of the weights of the arcs in the morphotactics transducer are found by setting the weights proportional to the number of traversals of each arc. We can use a specialized semiring to cleanly and efficiently count the number of traversals of each arc.

Weights in finite-state transducers are elements of a semiring, which defines two binary operations $\otimes$ and $\oplus$, where $\otimes$ is used to combine the weights of arcs on a path into a path weight and $\oplus$ is used to combine the weights of alternative paths (Berstel and Reutenauer, 1988). We define a counting semiring to keep track of the number of traversals of each arc. The weights in the mt transducer are converted to the counting semiring. In this semiring, the weigths are vectors of integers having dimension as the total number of arcs in the mt transducer. We number the arcs in the mt transducer and set the weight of the n^{th} arc as the n^{th} basis vector. The binary plus $\oplus$ and the times $\otimes$ operations of the counting semiring are defined as the sum of the weight vectors. Thus, the n^{th} value of the vector in the counting semiring just counts the appearances of the n^{th} arc of mt in a path.

To estimate the weights of the stochastic model of the mt transducer, we use the text corpus collected from the web. First we parse the words in the corpus to get all the possible analyses of the words. Then we disambiguate the morphological analyses of the words to select one of the morphosyntactic tag sequences x_i for each word. We build a finite-state transducer $\epsilon \times x_i$ that maps ϵ symbol to x_i in the counting semiring. The weights of this transducer are zero vectors having the same dimension as the mt transducer. Then the finite-state transducer $(\epsilon \times x_i) \circ (mt \times \epsilon)$ having all $\epsilon : \epsilon$ arcs can be minimized to get a one-state FST which has the weight vector that keeps the number of traversals of each arc in mt. The weight vector is accumulated for all the x_i morphosyntactic tag sequences in the corpus. The final accumulated weight vector is used to assign probabilities to each arc in the mt transducer proportional to the traversal count of the arc, hence resulting in the stochastic morphotactics transducer $\tilde{mt}$. We use add-one smoothing to prevent the arcs having zero probability. The $\tilde{mt}$ transducer is composed with the morphophonemics transducer mp to get a stochastic morphological parser.

The stochastic parser now returns probabilities with the possible analyses of a word. Figure 1 shows the weighted paths for the four possible analyses of the word *kedileri* as represented in the stochastic parser. The weights are negative log probabilities.

4 Spelling Correction

The productive morphology of Turkish allows one to generate very long words such as

ölümsüzleştirdiğimizden. Therefore, the detection and the correction of spelling errors by presenting the user with a ranked list of spelling suggestions are highly desired. There have been some previous studies for spelling checking (Solak and Oflazer, 1993) and spelling correction (Oflazer, 1996). However there has been no study to address the problem of ranking spelling suggestions. One can use a stochastic morphological parser to do spelling checking and correction, and present spelling suggestions ranked with the parser output probabilities. We assume that a word is misspelled if the parser fails to return an analysis of the word. Our method for spelling correction is to enumerate all the valid and invalid candidates that resemble the incorrect input word and filter the invalid ones with the morphological parser.

To enumerate the alternative spellings for a misspelled word, we generate all the words in one-character edit distance with the input word, where we consider one symbol insertion, deletion or substitution, or transposition of adjacent symbols. The Turkish alphabet includes six special letters (*ç*, *ğ*, *ı*, *ö*, *ş*, *ü*) that do not exist in English. These characters may not be supported in some keyboards and message transfer protocols; thus people frequently use their nearest ASCII equivalents (*c*, *g*, *i*, *o*, *s*, *u*, respectively) instead of the correct forms, e.g., spelling *nasılsın* as *nasilsin*. Therefore, in addition to enumerating words in one edit distance, we also enumerate all the words from which the misspelled word can be obtained by replacing these special Turkish characters with their ASCII counterparts. For instance, for the word *nasilsin*, the alternative spellings *nasılsin*, *nasilsın*, and *nasılsın* will also be generated.

Note that although the context is important for spelling correction, we use only unigrams. One can build a morpheme based language model to incorporate the context information. We also limited the edit distance to 1, but it is straightforward to allow longer edit distances. We can build a finite-state transducer to enumerate and represent efficiently all the valid and invalid word forms that can be obtained by these edit operations on a word. For example, the deletion of a character can be represented by the regular expression $\Sigma^*(\Sigma : \epsilon)\Sigma^*$ which can be compiled as a finite-state transducer, where Σ is the alphabet. The union of the transducers encoding one-edit distance operations and the restoration of the special Turkish characters is precompiled and optimized with determinization and minimization algorithms for efficiency. A misspelled input word transducer can be composed with the resulting transducer and in turn with the morphological parser to filter out the invalid word forms. The words with their estimated probabilities can be read from the output transducer and constitute the list of spelling suggestions for the word. The probabilities are used to rank the list to show to the user. We also handle the spelling errors where omission of a space character causes joining of two correct words by splitting the word into all combinations of two strings and checking if the string pieces are valid word forms. An example list of suggestions with the assigned negative log probabilities and their English glosses for the misspelled word *nasilsin* is given below.

nasılsın (14.2) (How are you), *nakilsin* (15.3) (You are a transfer), *nesilsin* (21.0) (You are a generation), *nasipsin* (21.2) (You are a share), *basilsin* (23.9) (You are a bacillus)

On a manually chosen test set containing 225 correct words which have relatively more complex morphology and 43 commonly misspelled words, the Precision and the Recall scores for the detection of spelling errors are 0.81 and 0.93, respectively.

5 Morphology-based Language Modeling

The closure of the transducer for the stochastic parser can be considered as a morphology-based unigram language model. Different than standard unigram word language models, this morphology-based model can assign probabilities to words not seen in the training corpus. It can also achieve lower out-of-vocabulary (OOV) rates than models that use a static vocabulary by employing a relatively smaller number of root words in the lexicon.

We compared the performances of the morphology-based unigram language model and the unigram word language model on a broadcast news transcription task. The acoustic model uses Hidden Markov Models (HMMs) trained on 183.8 hours of broadcast news speech data. The test set contains 3.1 hours of speech data (2,410 utterances). A text corpus of 1.2 million words from the transcriptions of the news recordings was used to train the stochastic parser as explained in Section 3 and unigram word language models.

We experimented with four different language

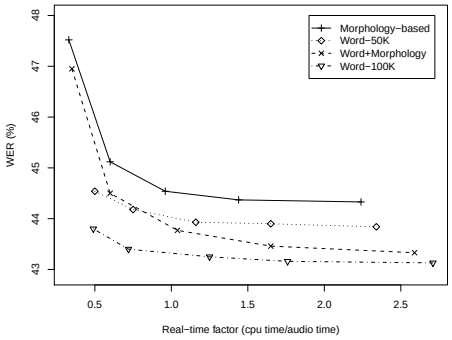

Figure 2: Word error rate versus real-time factor obtained by changing the pruning beam width.

models. Figure 2 shows the word error rate versus run-time factor for these models. In this figure the Word-50K and Word-100K are unigram word models with the specified vocabulary size and have the OOV rates 7% and 4.7% on the test set, respectively. The morphology-based model is based on the stochastic parser and has the OOV rate 2.8% . The 'word+morphology' model is the union of the morphology-based model and the unigram word model.

Even though the morphology-based model has a better OOV rate than the word models, the word error rate (WER) is higher. One of the reasons is that the transducer for the morphological parser is ambiguous and cannot be optimized for recognition in contrast to the word models. Another reason is that the probability estimates of this model are not as good as the word models since probability mass is distributed among ambiguous parses of a word and over the paths in the transducer. The 'word+morphology' model seems to alleviate most of the shortcomings of the morphology model. It performs better than 50K word model and is very close to the 100K word model. The main advantage of morphology-based models is that we have at hand the morphological analyses of the words during recognition. We plan to train a language model over the morphological features and use this model to rescore the hypothesis generated by the morphology-based models on-the-fly.

6 Conclusion

We described the first stochastic morphological parser for Turkish and gave two applications. The first application is a very efficient spelling correction system where probability estimates are used for ranking misspelling suggestions. We also gave the preliminary results for incorporating the morphology as a knowledge source in speech recognition and the results look promising.

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 and TÜBİTAK BİDEB 2211.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *CIAA 2007*, volume 4783 of *LNCS*, pages 11–23. Springer. `http://www.openfst.org`.

Jean Berstel and Christophe Reutenauer. 1988. *Rational Series and their Languages*. Springer-Verlag.

Jason Eisner. 2002. Parameter estimation for probabilistic finite-state transducers. In *ACL*, pages 1–8.

Tunga Güngör. 1995. *Computer Processing of Turkish: Morphological and Lexical Investigation*. Ph.D. thesis, Boğaziçi University.

Lauri Karttunen and Kenneth R. Beesley. 1992. Two-level rule compiler. Technical report, Xerox Palo Alto Research Center, Palo Alto, CA.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *ACL*, pages 178–181.

Kemal Oflazer and Sharon Inkelas. 2006. The architecture and the implementation of a finite state pronunciation lexicon for Turkish. *Computer Speech and Language*, 20(1):80–106.

Kemal Oflazer. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing*, 9(2):137–148.

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *GoTAL 2008*, volume 5221 of *LNCS*, pages 417–427. Springer.

Ayşin Solak and Kemal Oflazer. 1993. Design and implementation of a spelling checker for turkish. *Literary and Linguistic Computing*, 8(3):113–130.