Turkish handwritten text recognition: a case of agglutinative languages

Berrin Yanikoglu and Alisher Kholmatov

Sabanci University, Orhanli, Istanbul, Turkey 34956

ABSTRACT

We describe a system for recognizing unconstrained Turkish handwritten text. Turkish has agglutinative morphology and theoretically an infinite number of words that can be generated by adding more suffixes to the word. This makes lexicon-based recognition approaches, where the most likely word is selected among all the alternatives in a lexicon, unsuitable for Turkish. We describe our approach to the problem using a Turkish prefix recognizer. First results of the system demonstrates the promise of this approach, with top-10 word recognition rate of about 40% for a small test data of mixed handprint and cursive writing. The lexicon-based approach with a 17,000 word-lexicon (with test words added) achieves 56% top-10 word recognition rate.

Keywords: handwriting recognition, OCR, Turkish, agglutinative

1. INTRODUCTION

Off-line handwriting recognition is the task of recognizing the image of a handwritten text, in contrast to on-line recognition where the dynamic characteristics of the writing are available as well (as is the case with the pen input of handheld devices). In writing recognition, language modelling can be done in order to improve the results of the character recognition module. Typically, a lexicon containing the vocabulary for a given task is used to restrict the word hypothesis to be one of the allowed alternatives. One such example is the roughly hundred-word lexicon of numbers used for bank check processing. Using a lexicon greatly improves the word recognition rate, since it is often not the case that the character recognition module returns the (letters of the) correct word. This is especially true for cursive handwritten text. For narrow domains such as bank check processing of legal amounts, the approach is not only useful but essential.

Turkish is an agglutinative language where new words are formed by adding suffixes to the end of root words. There are grammatical rules governing which suffixes may follow which other, and in what order, but the number of possible words that may be generated by adding more suffixes is practically infinite. As such, a finite-size lexicon for Turkish would miss a significant percentage of Turkish words, including common ones (e.g. verb conjugations typicaaly have several suffixes added to the root form). This makes lexicon-based text recognition approaches unsuitable for Turkish, or other agglutinative languages.

In this paper, we describe a system to recognize unconstrained Turkish handwritten text. We use the term text here to mean a line of text, though the system described here is part of a working system for whole text documents. In order to deal with the lexicon problem, we use a morphological parser for Turkish that verifies whether a given string is a Turkish prefix,[1] in the word recognition stage. We will refer to this parser shortly as a prefix parser/recognizer, in this context. The word recognition algorithm is a modified Viterbi algorithm with beam search, such that the forward pass expands the most likely prefixes seen thus far, eliminating non-Turkish prefixes, as decided by the parser, altogether.

The system uses the segmentation and letter recognition algorithms previously developed for *English* text recognition and previously described elsewhere.[2,3] The use of the English letter recognizer for Turkish is described in Section 4.1. The rest of the paper is organized as follows: in sections 2 and 3, Turkish morphology and previous work related to this problem are presented. In Sections 4 and 5, we describe our system and experiments to test the system performance. Finally, Section 6 summarizes the results and discusses future work.

E-mail: {berrin,alisher}@sabanciuniv.edu.

2. TURKISH MORPHOLOGY

Turkish is an Ural-Altaic language, having agglutinative morphology: Turkish word forms consist of morphemes concatenated to a root morpheme or to other morphemes. Except for a very few exceptional cases, the surface realizations of the morphemes are conditioned by various morpho-phonemic processes such as vowel harmony, vowel and consonant elisions. There are about 120 suffixes and the number of forms one can derive from a root form may be in the millions!

As an example, the word "yap-abil-ecek-diy-se-niz", meaning "if you were going to be able to do" has five suffixes added to the root verb "yap". This is not an uncommon word artificially constructed to demonstrate the poin, but a typical verb conjugation. This example may indicate how big a general-purpose Turkish lexicon would need to be, if we were to use a simple lexicon-based approach for recognition, given that there are more than 100 suffixes. In fact, as mentioned by Carki et al.,[4] even a 500,000-word lexicon misses more than about 5% of Turkish words, encountered in the national newspapers.

3. PREVIOUS WORK

Much work has been done in off-line handwriting recognition for English.[3,5–12] The word recognition rates for unconstrained handwriting recognition systems can range from about 50% for systems using large lexicons, to more than 90% in restricted-vocabulary domains (check amounts, dates etc.). Due to differing data sets and varying sizes of lexicons, a direct comparison of systems is not made, though they typically use Hidden Markov Models (HMM) or some other form of dynamic programming approach to handle the segmentation ambiguity.

On the other hand, we don't know of any published research on recognizing Turkish text. The work of Carki et al. on Turkish speech recognition however relates to ours. They address the same problem as we do, namely the inadequacy of a finite-size lexicon for Turkish, but for the speech recognition task.[4] They report an out-of-vocabulary rate (OOV) of 15% for a 64K-word lexicon typically used for Turkish speech recognition, and an OOV rate of more than 5% for a 500K-word lexicon: in other words, 15% of words encountered during testing were not covered by the 64K-word lexicon. In order to reduce the OOV rate, they took an alternative approach and used a syllable-based lexicon. This indeed reduced the OOV rate, as expected; however the word error rate increased for their speech recognition system, since the new recognition units, namely syllables, suffers from higher (acoustic) confusability due to their shorter lengths.

Similar results would be expected in handwriting recognition systems using a syllable-based lexicon. Instead, using a prefix parser as described in the following sections, we are able to reduce the out-of-vocabulary rate to near zero, while avoiding the shortcomings of lexicons that are based on smaller units. The Turkish prefix parser/recognizer is used to determine whether a given letter string is the prefix of any Turkish word. It has about 60,000 root words, 35,000 of which are proper names (including some foreign names which may be inflected by Turkish suffixes) and returns true if a given string is the prefix of a Turkish word.

4. SYSTEM DESCRIPTION

The Turkish handwriting recognition system described in this paper is modified from our handwriting recognition system for English,[2,3] by changing only the letter recognition and word recognition subsytems. Unchanged components are described here shortly to provide an overview:

The preprocessing algorithm corrects the page skew, found using the Hough transform. Line boundaries are first located approximately between the peaks and valleys in the horizontal projection profile of the page, and then refined by a contour-following algorithm that can separate overlapping lines. After finding the baselines usign a heuristic similar to the one used in line finding, statistics defining the style of the writing, such as slant, pen-thickness, expected character size etc., are gathered, to be used in the segmentation and recognition algorithms. Word boundaries are decided using a threshold obtained by analyzing the histogram of spaces between connected components, guided by the expected character width information.

Words are then *over*-segmented using straight lines, by a heuristic where individual weights, such as how important it is to cut the word near the baseline, were decided using linear programming.[3] The segmentation algorithm divides a letter into at most three segments; so when recognizing letters, one, two and three-segments

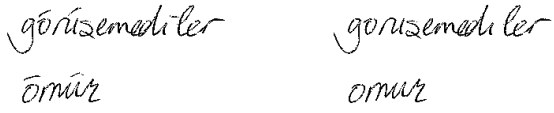

Figure 1. Effects of accent removal: input words on the left side of the image are shown on the right, after accent removal.

long input has been passed on to the letter recognition engine (resulting roughly in $3n$ calls to the letter recognizer for a word segmented into n segments). The number of segments used in the letter is taken into account in the word decoding phase. The letter recognition engine is a one-hidden layer feed-forward neural network, trained with a small data set of lowercase English characters, written by several writers.

4.1. Character Recognition

The modern Turkish alphabet, adopted in 1928, is based on the Latin alphabet. It differs from the English alphabet by the addition of six characters (ç, ş, ğ, ı, ö, ü) and omission of three others (q, w and x). Hence the 29-letter modern Turkish alphabet is: "a b c ç d e f g ğ h ı i j k l m n o ö p r s ş t u ü v y z".

As a first step to develop a Turkish recognition system without the laborious task of collecting and labeling character data, we built the Turkish character recognition engine from the neural network based character recognizer for English. The output for the Turkish system is the same as the English OCR engine for characters common in the two languages (a,b,c,d,e,f,...). For the remaining characters, we take the output of the most visually similar characters in the English alphabet: outputs of 'g', 'i', 'o', 's' and 'u' are assigned to 'ğ', 'ı', 'ö', 'ş', and 'ü', respectively, and the maximum of the outputs of 'c' and "q' is assigned to 'ç'. This mapping reflects the most likely outcome of our English OCR engine, given the Turkish characters. As a result of this shared output, the outputs of some letter pairs, for instance 'o' and 'ö', are the same in the new system.

In fact, most separate diacritical marks are discarded for simplicity during the image processing phase, avoiding the process of locating which diacritics go with which letter. This was also the case with our English handwriting recognition system, where the dots of the letters 'i' and 'j' were discarded to avoid their correct assignment problem. As a result, the Turkish letters 'ğ', 'ı', 'ö', and 'ü' lose their often separately written diacritics and are indistinguishable from the corresponding English letters 'g', 'i', 'o', 'u'; hence shared network outputs is reasonable. The letters 'ç' and 'ş', on the other hand, are often written with attached diacritics, and they reduce the system performance, since a 'ç' looks something between a 'c' and a 'q', for instance.

Eliminating diacritical marks simplifies the system and enables better use of the English letter recognizer, however it is not completely harmless. The input word *görüşemediler* becomes *gorusemedıler*, for instance, as shown in Figure 1, and both word hypotheses are equally likely in terms of the letter recognition results. However, the necessary disambiguation can often be done automatically using the redundancy provided by the Turkish wovel harmony, without using a lexicon. Simply speaking, Turkish vowel harmony dictates which letters can be used together in the same word (e.g. 'ö' can be used with 'e', 'i' and 'ü' in the same word, but not with 'a' or 'ı'). However, some words, such as the second input word *ömür* in Figure 1.a, maps to other valid Turkish words (*omur*) and the correct word cannot be identified without semantic processing.

To measure the potential degradation caused by the removal of diacritics, we replaced the accented Turkish characters in a 19000 word lexicon, with their closest English counterpart (e.g. as is done when Turkish is typed using a Western keyboard). About 1500 words (8%) were mapped to some other Turkish word (usually short words that are in root form where vowel harmony could not help). The disambiguation of these words would need to be handled using the diacritics information or contextual information. However the effects of our preprocessing are not that drastic because the accents of certain characters are not removed.

We are in the process of building a database of Turkish characters in order to train a Turkish OCR engine from scratch, using the system described in this paper to semi-automatically label the collected data.

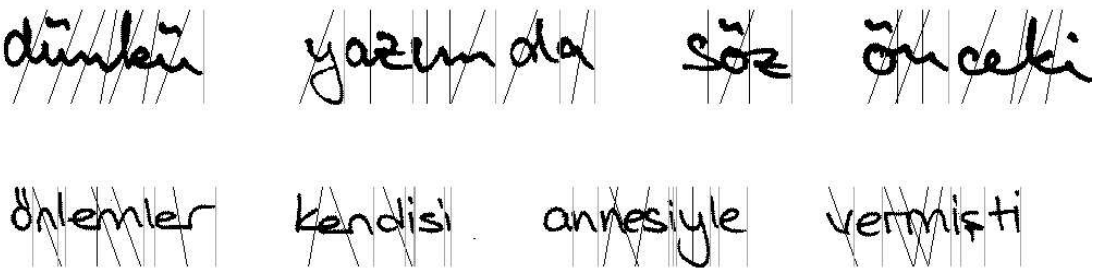

Figure 2. Sample segmentation output. Lighter lines indicate the natural letter boundaries.

4.2. Word Recognition

Word recognition is done on a word which is over-segmented as described in previous papers.[3] The segmentation output for two lines of the current test data is shown in Figure 2.

Since the segmentation algorithm divides a letter into at most three segments, each one, two and three segments-long input is passed to the letter recognition engine. For instance, an 'm' divided into three segments is presented to the letter recognizer in three forms, only one of which is the full letter; the corresponding character outputs of the letter recognizer may be an 'r', and 'n' and an 'm', for 1, 2 or 3 segment-long input, respectively.

Given n presegmentation points determining the observation sequence $O = O_1O_2O_3...O_n$ and the output of the neural network OCR engine for all one to three segments-long input (about $3n$ of them), the word recognition algorithm tries to find the most likely word hypothesis using a modified Viterbi algorithm with Beam search. Associated with each time frame is a stack of ordered prefixes. At each time frame, the modified Viterbi algorithm expands the best N most likely prefixes (partial paths) from previous three frames, eliminating those that are rejected by the parser, altogether. The number N is decreased over time to around 100, since more of the shorter prefixes need to be expanded (many of the two-letter strings are valid prefixes for instance). Currently, different segmentations of the word are considered equally likely, and the partial paths are ordered only by the network outputs being used as the posterior probabilities of letters given the input ($P(letter|O_i)$, $P(letter|O_{i,i+1}$ etc.). Finally, all the valid prefixes on the stack associated with the nth frame are checked to see if they are valid *words* (using a slightly different version of the parser), and the remaining word at the top of the stack is chosen as the correct word.

The algorithm can be outlined as follows:

1. for each time frame $t = 1$ to n
2. for each step size $i = 3$ to 1
   - if $t - i \geq 0$
     - expand the best paths at time $t - i$ using the neural network output for the input $(O_{t-i}...O_t)$
     - add the obtained partial paths to the stack of possible prefixes for time frame t
   - call the prefix parser to verify the partial paths for time t
   - sort the unrejected prefixes

The algorithm is illustrated partially in Fig. 4. Given the input word "gol", segmented into 4 segments as shown in Fig. 3, the 9 input pieces $(O_1, O_2, O_3, O_4, O_{1-2}, O_{1-3}, O_{2-3}, O_{2-4})$ are passed to the letter recognizer. The top part of Fig. 4 shows the best partial paths and the ones that are rejected by the prefix parser, while the bottom part shows the input passed on to the neural network at each step (path taken). The Turkish prefix

Figure 3. Sample word and its segmentation.

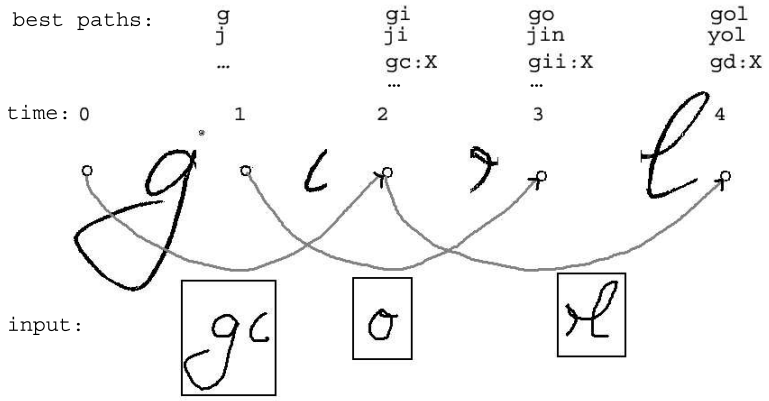

Figure 4. Illustration of the modified Viterbi decoding with Beam search, using the prefix parser. The rejected prefixes are marked with an X.

parser is used to determine whether a given letter string is the prefix of any Turkish word. In this example, the partial paths giving the prefixes "gc", "gii" and "gd" are rejected as there are no words starting with these prefixes.

For comparison, we also used a lexicon-based approach with a 17,000-word lexicon obtained from Milliyet (one of the prominent Turkish newspapers), expanded by the addition of the test words. In this approach, a standard Viterbi algorithm was used to match the segmented image to each word in the lexicon, while everything else was the same as the parser approach.

5. EXPERIMENTS

We tested the system using a 50-words excerpt taken from newspaper articles, excluding proper names, numbers and punctuation marks. The test data was written in lower-case by 9 writers whose writing were not in the training set, using their normal handwriting. Figure 5 shows some examples from the test data.

In all of the results, we report top-10 performance results for word recognition, since the system performance is still low. The performance of the lexicon-based approach for *handprint* data ranged between 22% and 75%, for top-10 word choices for different writers. On the average, about 57% of all the test words were in the top 10 choices of the word recognizer. The performance of this approach for *cursive* data (unconstrained: including run-on, connected and separate letters) ranged from 38-70%, with an average of 53% for top-10 performance.

The performance figures using the Turkish-prefix parser were lower: averaging 41% for handprint data and 38% for cursive writing. Even though the performance results are lower, this is to be expected: the parser recognizes potentially an infinite number of words, generating many words that are morphologically correct but semantically meaningless/unlikely, for the sake of covering all Turkish words. This means that there are more words that compete with the correct word, reducing the word recognition rate.

In order to deal with the unlikely words accepted by the prefix parser, we added a trigram language model to reorder the final word hypotheses. However, it did not improve the recognition results significantly since the competing unlikely words (which increase the effective size of the lexicon to at least a few hundred thousand) often do have high trigram probabilities, being morphologically correct.

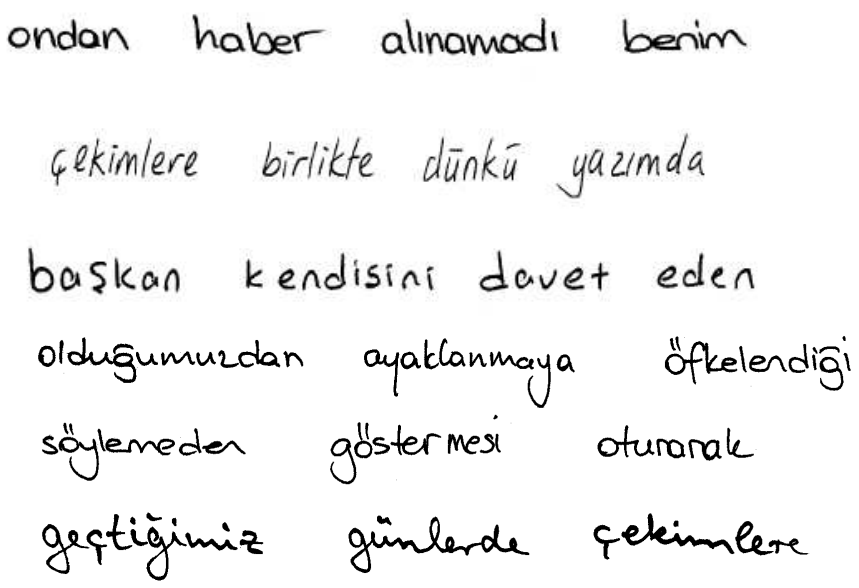

Figure 5. Sample test data.

Top ten recognition rates for unconstrained English handwritten text, reported by Marti and Bunke, is 67% with a small (7719 words) dictionary.[7] Even though these results are not directly comparable, it indicates that our results are promising.

6. SUMMARY AND FUTURE WORK

We have described a system to recognize Turkish handwritten text, built from our offline, cursive handwriting recognition system for English, by only modifying the character and word recognition processes. The approach taken for word recognition can also be used for other agglutinative languages, given a prefix parser for that language.

The word recognition performance ranged from 53% to 57% depending on the writing type for the lexicon-based approach and between 38% to 41% for the parser-based approach. These performance figures are not very impressive, but they show promise given that the OCR engine was not trained with Turkish characters.

The system is currently limited in that it can only recognize lowercase characters (no punctuation characters). However it is readily applicable to machine print text. The machine print recognition algorithm constrains the segmentation to be done by segmenting the word via vertical lines only (assuming no kerning). Recognizing machine print is easier since machine-print characters have less shape variance and it is easier to locate the reference lines, improving many components of the system.

Future work will include training the character recognition engine with Turkish characters, as well as testing the system with a larger test set.

7. ACKNOWLEDGEMENTS

We would like to thank Dr. Kemal Oflazer for providing us with the morphological analyzer and his help on the topic.

This project has been funded by TÜBİTAK, the Scientific and Technical Research Council of Turkey (project code 101E012).

REFERENCES

1. K. Oflazer, "Two-level description of Turkish morphology," *Literary and Linguistic Computing* **9**(2), 1994.
2. B. Yanikoglu, *Segmentation and Recognition of Off-line Cursive Handwriting.* PhD thesis, Dartmouth College, 1993.
3. B. Yanikoglu and P. A. Sandon, "Segmentation of off-line cursive handwriting using linear programming," *Pattern Recognition* **31**, pp. 1825–1833, 1998.
4. K. Carki, P. Geutner, and T. Schultz, "Turkish lvcsr: Towards better speech recognition for agglutinative languages," *Literary and Linguistic Computing* **9**(2), 2000.
5. Y. Tay, P. Lallican, M. Khalid, C. Viard-Gaudin, and S. Knerr, "An offline cursive handwritten word recognition system," *Proceedings of IEEE Region 10 Conference* , 2001.
6. T. Breuel, "Segmentation of handprinted letter strings using a dynamic programming algorithm," *Proc. of Sixth International Conference on Document Analysis and Recognition* , pp. 821–826, 2001.
7. U. Marti and H. Bunke, "Handwritten sentence recognition," *Proc. 15 th International Conference on Pattern Recognition* **3**, pp. 467–470, 2000.
8. X. Y. Ye, C. Y. Suen, and M. Cheriet, "A generic system to extract and clean handwritten data from business forms," *Proc. Int. Workshop on Frontiers in Handwriting Recognition* , pp. 63–72, 2000.
9. T. Steinherz, E. Rivlin, and N. Intrator, "Offline cursive script word recognition - a survey," *IJDAR* **2**(2-3), pp. 90–110, 1999.
10. C. Y. Suen, Q. Xu, and L. Lam, "Automatic recognition of handwritten data on cheques," *Pattern Recognition Letters* , pp. 1287–1292, 1999.
11. M.-Y. Chen, A. Kundu, and S. Srihari, "Variable duration hidden markov model and morphological segmentation for handwritten word recognition," *IEEE Trans. on Image Processing* **4**(12), pp. 1675–1688, 1995.
12. A. Senior, *Off-line cursive handwriting recognition using recurrent neural networks.* PhD thesis, University of Cambridge, 1994.