Combining Syntactic Knowledge and Visual Text Recognition: A Hidden Markov Model for Part of Speech Tagging In a Word Recognition Algorithm

Jonathan J. Hull
Center of Excellence for Document Analysis and Recognition
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York 14260 USA
hull@cs.buffalo.edu

Abstract

The use of a hidden Markov model (HMM) for the assignment of part-of-speech (POS) tags to improve the performance of a text recognition algorithm is discussed. Syntactic constraints are described by the transition probabilities between POS tags. The confusion between the feature string for a word and the various tags is also described probabilistically. A modification of the Viterbi algorithm is also presented that finds a fixed number of sequences of tags for a given sentence that have the highest probabilities of occurrence, given the feature strings for the words. An experimental application of this approach is demonstrated with a word hypothesization algorithm that produces a number of guesses about the identity of each word in a running text. The use of first and second order transition probabilities is explored. Overall performance of between 65 and 80 percent reduction in the average number of words that can match a given image is achieved.

1. Introduction

Text recognition algorithms often process only images of isolated characters. This is sometimes followed by a post-processing step that uses information from a dictionary of allowable words to correct recognition errors. This approach can provide high performance for good quality images.

A computational theory for word recognition has been proposed that overcomes some of the constraints of other methodologies [6]. The design of this technique is based on human performance in reading which suggests that the feature analysis of word images includes a wholistic analysis of word shape. Also, feature extraction from a word image is only one part of a complex process of developing an understanding of a text. Furthermore, the model suggests that to achieve high levels of performance in text recognition for a range of input qualities it may be necessary to understand the text as well. One part of the understanding process that underlies word recognition is an analysis of the syntax of the text.

This paper discusses aspects of a Markov model for POS tagging where the probability of observing any POS tag is dependent on the tag of the previous word [8,9]. This model is applied to text recognition by first using a word recognition algorithm to supply a number of alternatives for the identity of each word. The tags of the alternatives for the words in a sentence are then input to a modified Viterbi algorithm that determines sequences of syntactic classes that include each word. An alternative for a word decision is output only if its part of speech tag is included in at least one of these sequences. The Markov model improves word recognition performance if the number of alternatives for a word are reduced without removing the correct choice.

The rest of this paper briefly introduces the computational model for word recognition. This is followed by a description of how a Markov model for language syntax is incorporated in the model. The modified Viterbi algorithm proposed in this paper is then described. The performance of this technique in reducing the number of alternatives for words in a sample of text is then discussed. The effect of employing the first and second order Markov assumptions and different methods of estimating the probabilities are explored.

2. Computational Model

The word recognition algorithm that incorporates the Markov model of syntax contains the three steps shown in Figure 1. The input is a sequence of word images $w_i, i = 1,2,\cdots$. The hypothesis generation stage computes a group of possible identifications for w_i (called N_i or its *neighborhood*) by matching a feature representation of the image to the entries in a dictionary. The global contextual analysis phase uses information about other words that have been recognized, such as their syntactic classification, to constrain the words that can be in N_i. The output is a neighborhood N_i^* of reduced size. The hypothesis testing phase uses the contents of N_i^* to determine a specific set of feature tests that could be executed to recognize w_i. The output of hypothesis testing is either a unique recognition of w_i or a set of hypotheses that contains the word in the image.

3. Syntax Model

The syntax of a sentence is summarized as the sequence of syntactic classifications for its words. A specific sequence of syntactic classes is referred to as a "parse" for the sentence. For example, in the sentence "He was at work.", *He* is a pronoun, *was* is a verb, *at* is a preposition, and *work* is a noun. Since it is known that the appearance of any syntactic class probabilistically constrains the classes that can follow it, a Markov model is a natural representation for syntax [11]. An example of such a probabilistic constraint is given by the probabilities that certain syntactic classes follow an article in a large sample of text. The word following an article is a singular or mass noun in 51 percent of all cases and is an adjective 20 percent of the time. The other 29 percent of occurrences are scattered over 82 other syntactic classes [4].

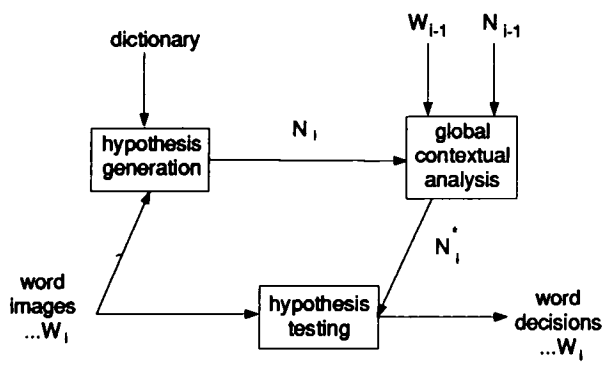

Figure 1. The word recognition algorithm.

A hidden Markov model (HMM) can be specified that links the recognition process described earlier and a Markov model for POS tag assignment [12]. The grammatical classes in the English language are assumed to be the N *states* of a discrete n^{th} order Markov process. In the word recognition algorithm, the states are "hidden" because they are not observable at run-time. Rather, the feature vector that describes a word is the observable event. The number of such events is finite and provides a fixed number M of *observation symbols.*

The transition from one state to another is described by a *state transition probability distribution.* If the Markov process is assumed to be first order, this distribution can be given be an $N \times N$ matrix. A second order assumption would imply the use of an $N \times N \times N$ probability distribution matrix.

There is also a probabilistic constraint on the appearance of an observation or feature vector given that the model is in a specific state. For example, if the syntax specifies that an *article* should occur, the feature vector would describe the word *the* with a certain probability and the word *a* with another probability. This constraint is sometimes referred to as the *confusion probability* that an observation occurs given that the process is in a specific state.

There is also an *initial state distribution* that specifies the state or grammatical class that the model is in for the first word in a sentence. This initial constraint can be powerful. For example, it has been observed in a sample of newspaper reportage that the first word in a sentence is an article or a proper noun with probability 0.31 and 0.14, respectively. The other 55 percent is divided among 24 other classes.

The HMM is completely specified by the five elements just described (states, observation symbols, state transition probabilities, observation symbol probabilities, and initial probabilities). The HMM is applied to word recognition by estimating the sequence of states (grammatical classes) with the maximum a-posterior probability of occurrence for a given sequence of observations (feature vectors). Each observation can correspond to a number of different words with a similar feature vector (i.e., the neighborhood in the word recognition algorithm). The performance of word recognition is improved by removing words from neighborhoods that have syntactic classes which do not occur on the

estimated state sequence.

The limitation to a single best state sequence could be too constraining for this application. Rather, it is desired to locate a number of state sequences that have a higher a-posterior probability of occurrence than any other state sequences. This would allow for a word image to potentially be constrained by more than one syntactic class, thus loosening the syntactic constraint.

The estimation of the sequence of states with the maximum a-posterior probability of occurrence is efficiently performed by the Viterbi algorithm [3]. The adaptation of the Viterbi algorithm to this problem is very similar to its use in postprocessing character decisions [5]. The Viterbi algorithm has also been successfully used for speech recognition [1].

Under the assumptions that sentence delimiters (periods) are perfectly recognized and the occurrence of a word recognition error in any position is independent of the occurrence of an error in any other position, the Viterbi algorithm determines the string of syntactic classes $\bar{Z} = z_0, z_1, \ldots, z_{n+1}$ that maximize Bayes' formula:

$$P(\bar{Z} \mid \bar{X}) = \frac{P(\bar{X} \mid \bar{Z})\, P(\bar{Z})}{P(\bar{X})} \qquad 1$$

where $\bar{X} = x_0, x_1, \ldots, x_{n+1}$ is the sequence of feature vectors for the words in the sentence and $x_0 = x_{n+1} = z_0 = z_{n+1} = period$.

The independence and Markov assumptions, as well as the fact that the maximization of equation 1 is independent of $\bar{X}$ reduces the maximization of equation 1 to the maximization of:

$$\prod_{i=1}^{n+1} P(X_i \mid Z_i)\, P(Z_i \mid Z_{i-1}) \qquad 2$$

over all possible $\bar{Z}$, where $P(X_i \mid Z_i)$ is the conditional probability of the feature vector X_i taking on its value given that the corresponding syntactic class is Z_i. This is referred to as the observation symbol probability or confusion probability. $P(Z_i \mid Z_{i-1})$ is the first order transition probability of the occurrence of syntactic class Z_i given that syntactic class Z_{i-1} has been observed. To avoid the combinatorial explosion inherent in calculating equation 2 for all possible strings $\bar{Z}$, the Viterbi algorithm uses a dynamic programming formulation to transform the problem into one of graph searching with a computational requirement on the order of M^2 operations, where M is the number of alternatives for any word image.

A useful modification of the Viterbi algorithm is to allow it to find a fixed number of POS sequences (parses) for a sentence, each of which have the next best cost among all possible alternatives. This modification is simply implemented by maintaining the desired number of alternative sequences and their costs at each point in the evaluation. The final result includes the parses and their costs.

4. Experimental Investigation

Experimental tests were conducted to determine the ability of the HMM to reduce the number of word candidates that match any image. Given sentences from test samples of running text, a set of candidates for each word were produced by a model for the hypothesis generation portion of the word recognition algorithm. These candidates were looked up in a dictionary to retrieve their syntactic classes as well as their confusion probabilities. The Viterbi algorithm was then run on these data, using transition probabilities from another large text. Word candidates were removed from a neighborhood if their syntactic classes did not appear in any of the results produced by the Viterbi.

Performance was measured by calculating the average neighborhood size per text word before and after the application of syntax. This statistic is defined as:

$$ANS_t = \frac{1}{N_w} \sum_{i=1}^{N_w} ns_i$$

where N_w is the number of words in the test sample and ns_i is the number of words in the neighborhood for the i^{th} word in the text. The improvement in performance is measured by the percentage reduction in ANS_t. The *error rate* was also measured. This is the percentage of words with neighborhoods that do not contain the correct choice after the application of syntax.

4.1. Experimental Design

Experiments were designed to explore several questions about the application of the HMM.

The effect of accuracy in neighborhood calculation was determined by applying two alternative models for hypothesis generation. One model produces larger neighborhoods than the other. The objective was to determine the extent to which the effect of syntax degrades as neighborhood size increases. It was expected that as more choices occurred in the neighborhood for a word and the number of alternative syntactic classes increased, that the ability of the HMM to locate the correct word candidates would decrease. This is important to know because in an application it may be necessary to consider somewhere between five to thirty choices for a word to obtain a high correct rate in neighborhood calculation [7]. Ideally, the HMM would

significantly reduce neighborhoods of any magnitude with an acceptable error rate.

The effect of using different numbers of parses for a sentence was also explored. As the number of parses was increased, the amount of applicable syntactic information increased. This should inhibit the reduction in neighborhood size because more alternatives are possible for any word. However, the error rate should also decrease for similar reasons. In the best case, some number of parses can be found that provide a significant reduction in neighborhood size with a low error rate.

The tradeoffs in using first-order versus second-order transition probabilities were also determined. It was expected that better performance would be obtained with second-order probabilities since they would better estimate the effect of language syntax. However, the question of whether to use second order probabilities is balanced by the implied storage cost and potential difficulty in obtaining accurate estimates for them.

Different methods for estimating the transition probabilities and their effect on performance were also explored. This is a significant question in such a system since the probabilities are typically estimated from a finite sample of text and it may be difficult to accurately estimate the probability of infrequently occurring transitions [2]. It is interesting that such transitions that can provide strict syntactic constraints since when they actually occur in a previously unseen sample of text, they could tightly constrain the identity of the corresponding words. However, if the syntactic transition has never been observed, the correct word could be eliminated from the neighborhood. That is, an error would occur.

Several methods for calculating transition probabilities and their effect on performance were explored. Each method requires that counts first be calculated of the number of occurrences of syntactic class transitions in a corpus of text. It is necessarily assumed each word in the training corpus is tagged with its grammatical class. The Bayesian estimate for the transition probabilities was one of those that were tested:

$$P(B\,|A) = \frac{c_{B\,|A} + 1}{n_A + 2} \qquad 3$$

where $c_{B\,|A}$ is the number of transitions counted in the training corpus and n_A is the total number of occurrences of syntactic category A. The advantage of this approach is that it provides a small non-zero probability of occurrence for every transition.

A version of the maximum likelihood probability estimate was also used:

$$P(B\,|A) = \frac{ceiling\ (\ c_{B\,|A}\ ,\ thresh\ )}{n_A} \qquad 4$$

where,

$$ceiling\ (\ c_{B\,|A}\ ,\ thresh\ ) = thresh\ , \quad \text{if } c_{B\,|A} \le thresh$$

$$= c_{B\,|A}\ , \quad \text{otherwise.}$$

The purpose of using this estimate is to decrease the effect of infrequently occurring transitions by making all transitions that occur less than a fixed number of times equivalent to one another. It was expected that as *thresh* was increased, the ability of the HMM to reduce the average neighborhood size would decrease and the error rate would also decrease. This would occur because the effect of infrequently occurring transitions was being nullified with increases in *thresh*.

4.2. Text Database

A soft copy (ASCII) text sample known as the Brown Corpus was used for the experiments [10]. This text was chosen because it is large (over 1,000,000 words of running text) and every word is tagged with its syntactic class. The corpus is divided into 15 subject categories or genres. There are 500 individual samples of running text in the corpus and each one contains approximately 2000 words. The number of samples in each genre differs depending on the amount published in that area at the time the corpus was compiled.

The syntactic tags used to describe words in the corpus are organized in six major categories:

(1) *parts of speech:* nouns, verbs, adjectives, and so on.

(2) *function words:* determiners, prepositions, conjunctions, and so on.

(3) *important individual words: not*, existential *there*, infinitival *to*, forms of *do*, *be*, and *have*.

(4) *punctuation marks of syntactic importance:* ".", "(", ")", "--", ",", ":".

(5) *inflectional morphemes:* noun plurals, and possessives, verb past, present and past participle, and so on.

(6) *foreign or cited words.*

A tag sometimes has a 'U' affixed to it to indicate the word is a negation. Altogether, 84 different tags were used in the experiments described in this paper. Those tags are listed in Appendix 1.

4.3. Hypothesis Generation Algorithm

The operation of the hypothesis generation algorithm was simulated by calculating the feature description for a word from pre-defined features for the letters in the word. All the words in a dictionary with the same feature description were used as its neighborhood. Two feature descriptions that produce different neighborhoods were used to demonstrate the effect of neighborhood size on performance.

The feature descriptions are specialized for lower case characters because the experimentation is restricted to text written in lower case. The first feature description includes vertical bars of different heights, dots, and empty spaces. These features were chosen because they can be reliably computed from images of text even if the characters touch one another [6]. The complete listing of this feature set is given below:

1. A significant area at the beginning or end of a word that does not contain a vertical bar (e.g., the space to the right of the vertical bar in a "c" or an "e");
2. A short vertical bar (e.g., the leg of an "r");
3. A long high vertical bar that extends above the main bar of the word (e.g., the ascender portion of a "b");
4. A long low vertical bar that extends below the main body of the word (e.g., the descender in a "p");
5. Dots over short vertical bars (occurs in an "i");
6. Dots over long vertical bars (occurs in a "j").

When the feature description is applied to a word it yields a symbolic representation that would correspond to the sequence of occurrence of the features in an image of the word. Thus, both "me" and "may" have a symbolic representation 22221 and the neighborhood of "me" is {"me", "may"}.

The second feature set includes all the features in the first description plus the holes produced by topological containment. The addition of this feature provides a finer discrimination in neighborhood calculation, i.e., smaller neighborhoods.

4.4. Experimental Results

The HMM was applied to correct the simulated text recognition results produced by the two models for hypothesis generation described earlier. Five parses for each sentence in the test samples were output by the HMM and used to filter the neighborhoods. One test sample was randomly selected from each genre of the Brown Corpus. In each case, both first and second-order syntactic class transition probabilities were estimated from the remainder of the corpus. Both the Bayesian estimate (equation 3) and six values (*thresh* = 0,...,5) for the thresholded estimate (equation 4) were used.

The original values for ANS_t, before running the HMM averaged about 35 words for the first description and about 4.5 words for the second description.

The results of applying the HMM to sample A06 are shown in Table 1. The effect of using up to five different parses is shown by the columns numbered 1 to 5 at the top of the table. The top half of the table refers to the first feature description and the bottom half to the second. Each half also includes an indication of the two orders of transition probability that were used and the different methods for estimating them. The particular estimation procedure is coded by the effect it has on counts that originally were zero. Thus, 0.5 indicates the Bayesian estimate, 0 the thresholded estimate where *thresh* is 0, and so on.

The results show a number of interesting effects. It is seen that the first feature description, i.e., the one that produced original neighborhood sizes of about 35, produces the greatest reduction in ANS_t. Values between 80 and 85 percent are typical. However, error rates of between 9 and 14 percent occurred. The second feature description, i.e., the one that produced values of ANS_t of about four to five, provided more encouraging results. A 65 to 70 percent reduction in ANS_t was obtained with an error rate somewhere between one and two percent.

The modification to the Viterbi algorithm that located a number of alternative parses was also very successful at reducing the error rate with a negligible loss in reduction of ANS_t. The top five parses yielded error rates that were about two-thirds those of the first parse, for the first feature description. When the second feature description was used, the error rates were reduced to between one-third and one-half their previous values when the top five parses were applied.

The effect of using first-order versus second-order transition probabilities was almost negligible. The results show that the percentage reduction is virtually unaffected by the difference in syntactic information. The error rates show some minor differences, but nothing dramatic. For example, with the second feature description and using the Bayesian estimate, and five parses, the error rate was 1.2 with first-order information and 1.0 with second-order. However, on the whole, there were no remarkable differences.

			parses									
feature	trans	trans	1		2		3		4		5	
desc.	order	est.	ErrRate	%reduce	ErrRate	%reduce	ErrRate	%reduce	ErrRate	%reduce	ErrRate	%reduce
1	1	0.5	12.82	84.64	10.95	83.07	9.94	81.91	9.43	81.19	8.83	80.30
1	1	0	12.71	84.66	11.00	83.19	9.94	81.99	9.43	81.29	8.93	80.39
1	1	1	12.87	84.67	11.00	83.19	9.94	81.99	9.43	81.28	8.93	80.40
1	1	2	12.82	84.61	10.90	83.02	9.89	81.99	9.28	81.40	8.78	80.20
1	1	3	12.87	84.56	10.85	83.01	9.84	81.80	9.23	81.02	8.78	80.07
1	1	4	12.87	84.56	10.85	83.02	9.84	81.86	9.28	81.09	8.78	80.06
1	1	5	12.87	84.57	11.00	83.08	10.04	81.80	9.38	81.01	8.73	80.19
1	2	0.5	12.61	84.77	11.25	83.37	10.09	81.90	9.33	81.03	8.83	80.19
1	2	0	13.52	84.63	12.01	83.03	10.80	81.74	10.24	81.42	9.23	80.46
1	2	1	13.02	84.82	11.71	83.57	10.34	81.95	9.43	81.22	8.68	80.37
1	2	2	12.71	84.77	11.30	83.28	9.94	81.71	9.08	81.34	8.73	80.71
1	2	3	13.02	84.52	11.60	83.15	10.34	81.64	9.79	80.90	9.18	80.40
1	2	4	13.72	84.37	12.21	83.19	10.95	81.59	10.29	80.86	9.33	80.38
1	2	5	13.82	84.88	12.36	83.43	11.20	81.94	10.54	81.25	9.38	80.65
2	1	0.5	2.83	69.24	2.07	68.35	1.66	67.42	1.46	66.77	1.21	65.77
2	1	0	2.83	69.23	2.02	68.36	1.66	67.39	1.41	66.75	1.21	65.68
2	1	1	2.83	69.23	2.02	68.31	1.72	67.42	1.46	66.73	1.26	65.79
2	1	2	2.83	69.23	2.07	68.32	1.72	67.42	1.46	66.71	1.21	65.79
2	1	3	2.83	69.23	2.02	68.32	1.77	67.42	1.46	66.71	1.26	65.80
2	1	4	2.88	69.23	2.02	68.38	1.77	67.47	1.46	66.72	1.26	65.85
2	1	5	2.77	69.23	2.02	68.38	1.77	67.47	1.46	66.72	1.26	65.83
2	2	0.5	2.17	69.11	1.66	68.19	1.36	67.34	1.11	66.68	1.01	65.94
2	2	0	3.68	69.01	2.83	67.88	2.42	66.78	2.27	66.16	2.17	65.43
2	2	1	2.37	69.07	1.72	68.14	1.31	67.31	1.21	66.67	1.01	66.01
2	2	2	2.22	69.05	1.72	68.24	1.21	67.43	1.16	66.96	1.01	66.10
2	2	3	2.27	69.13	1.77	68.22	1.41	67.38	1.31	66.85	1.21	66.10
2	2	4	2.52	69.13	1.87	68.22	1.41	67.38	1.26	66.81	1.11	66.23
2	2	5	2.42	69.07	1.87	68.23	1.51	67.65	1.26	66.99	1.16	66.39

Table 1. Results of applying the HMM to sample A06.

The effect of the different estimation procedures for the transition probabilities is interesting. In all cases, the reduction in ANS_t is unaffected by the transition probability estimation technique. The error rate is also largely unaffected with the exception of the results for second order transition probabilities for the second feature description. The error rate for the thresholded estimate where *thresh* = 0 is between 50 and 100 percent higher than comparable values for the same number of parses. This suggests that there is significant negative information in the non-occurrence of certain transitions. However, the general trend of results suggests that this is best overcome by the maximum likelihood estimation procedure. The same experiments were also performed on one sample randomly selected from each genre. The results are comparable to those presented above.

5. Discussion and Conclusions

A hidden Markov model for applying language-level syntactic constraints in a text recognition algorithm was presented. The recognition algorithm produced a set of choices (referred to as a *neighborhood*) for each word in a sentence. Syntactic constraints were then used to remove some of these choices and thereby improve recognition performance.

Syntax was modeled as an n^{th} order Markov source and the Viterbi algorithm was used to find the sequence of syntactic classes that best fit the choices provided by the recognition algorithm. A modification of the Viterbi algorithm provided several next-best alternatives for the syntactic class sequence as well as their costs.

An experimental investigation of this approach was performed in which a simulation of two versions of the word recognition procedure were run on several samples of text under various conditions. It was shown that small initial neighborhoods, on the order of five choices per word, produced the best performance. Also, the modification of the Viterbi algorithm was quite valuable in reducing the error rate while also providing a reduction in the average neighborhood size of about 65 percent.

It was interesting that there was almost no difference in performance depending on whether first or second-order transitions were used. This is almost counter-intuitive but is still a valuable result since it suggests that a working implementation can be achieved with a small cost in storage. Also, there was almost no difference, except in one case, provided by the different methods for estimating the transition probabilities. This suggests that the technique can be sensitive to low frequency transitions. However, the specific circumstances in which this occurs are open to question and should be investigated further.

References

1. V. Cherkassky, M. Rao, H. Weschler, L. R. Bahl, F. Jelinek and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5*, 2 (March, 1983), 179-190.
2. K. Church, W. Gale, P. Hanks and D. Hindle, "Parsing, word associations, and typical predicate-argument relations," *International Workshop on Parsing Technologies*, Pittsburgh, Pennsylvania, August 28-31, 1989, 389-398.
3. G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE 61*, 3 (March, 1973), 268-278.
4. W. N. Francis and H. Kucera, *Frequency Analysis of English Usage: Lexicon and Grammar*, Houghton Mifflin, Co., Boston, Massachusetts, 1982.
5. J. J. Hull, S. N. Srihari and R. Choudhari, "An integrated algorithm for text recognition: comparison with a cascaded algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-5*, 4 (July, 1983), 384-395.
6. J. J. Hull, "Hypothesis generation in a computational model for visual word recognition," *IEEE Expert 1*, 3 (Fall, 1986), 63-70.
7. J. J. Hull, T. K. Ho, J. Favata, V. Govindaraju and S. N. Srihari, "Combination of segmentation-based and wholistic handwritten word recognition algorithms," *From Pixels to Features III: International Workshop on Frontiers in Handwriting Recognition*, Bonas, France, September 23-27, 1991, 229-240.
8. J. J. Hull, "Incorporation of a Markov model of language syntax in a text recognition algorithm," *Symposium on Document Analysis and Information Retrieval*, Las Vegas, Nevada, March, 1992.
9. J. J. Hull, "A hidden Markov model for language syntax in text reconition," *11th IAPR International Conference on Pattern Recognition*, The Hague, The Netherlands, August 30 - September 3, 1992.
10. H. Kucera and W. N. Francis, *Computational analysis of present-day American English*, Brown University Press, Providence, Rhode Island, 1967.
11. R. Kuhn, "Speech recognition and the frequency of recently used words: A modified Markov model for natural language," *Proceedings of the 12th International Conference on Computational Linguistics*, Budapest, Hungary, August 22-27, 1988, 348-350.
12. L. R. Rabiner and B. H. Huang, "An introduction to hidden Markov model," *ASSP Magazine 3*, 1 (1986), 4-16.