Adding Linguistic Constraints to Document Image Decoding

Kris Popat, Dan Bloomberg, and Dan Greene

Xerox Palo Alto Research Center, Palo Alto CA 94304, USA
{popat,bloomberg,greene}@parc.xerox.com

Abstract. A means of incorporating soft linguistic constraints in a document image decoding system is described. Document image decoding recognizes text by finding a most probable path through a hypothesized Markov source model for a given degraded document image. The linguistic constraints are expressed by a sequential predictive probabilistic language model. Search is accomplished by iteratively rescoring complete paths using conditional language model probability distributions of increasing order, expanding state only as necessary. This approach results in a solution that is provably optimal with respect to the specified source, degradation, and language models. Simulation results are presented for a recognition system wherein the documents are one-dimensional corrupted streams of American Morse Code pulses. This simulation preserves the essential features and challenges of text line decoding in a simplified setting that highlights the important algorithmic issues.

1 Best-Path Search in Document Image Decoding

Document image decoding (DID) [5, 6] is a method of text recognition in document images that is based on a communications systems view of the document composition, printing, degradation, and scanning processes. Among the advantages of DID are high recognition accuracy in situations where extensive customization is allowable, the ability to recognize some higher-level structure along with the text, and the ability to extend and improve the system within a consistent probabilistic framework. In the work on DID reported until now, the high recognition accuracy has been achieved despite a lack of any prior specification of which recognized strings are linguistically valid.

In DID, document images are modeled as having been produced by a Markov source (a probabilistic finite-state machine). The source begins in a distinguished *start* state and terminates in a distinguished *stop* state. Each transition causes the imaging of a character template (bitmap) on the page at a current cursor location, and advances that location in preparation for printing the next character. The character template may be whitespace. Formally, each transition in the source is assigned a four-tuple consisting of: a character template, a two-dimensional displacement by which to advance the cursor, the prior probability

[Submitted to *Proceedings of the 4th IAPR Workshop on Document Analysis Systems*, Rio de Janeiro, Brazil, December 2000.]

of following the transition, and a string label. Every complete path through the source thus defines a document image and an associated transcription: the image is the union of the bitmaps imaged on each transition, and the transcription is the concatenation of the associated string labels.

For a given observed document image, recognition involves finding a complete path through the hypothesized Markov source that best explains the observed image. Specifically, a complete path is sought that is most probable considering the entire image as evidence, where the probability is computed on the basis of the prior probabilities of the transitions and the likelihoods of the associated imaged templates, allowing for possible corruption of the imaged templates by an assumed probabilistic degradation process. Dynamic programming is used to find such a best path.

1.1 Separable Markov Sources and Text Line Decoding

In many instances, the document can be satisfactorily modeled by a two-level Markov source structure in which the top level characterizes and accounts for the vertical layout of a document page; the transitions at that level correspond to subsources describing individual text lines [4]. Such a source is termed *separable*. Because linguistic constraints bear most strongly on recognition within text lines, we focus on the decoding of individual text lines. Thus, the problem of interest is to find a best complete path through a subsource representing a text line.

In the absence of any linguistic constraints, a suitable subsource for text line decoding consists of a start state, a single interior state, and a stop state. The interior state has one self-transition for each character template. Control commences in the start state with the cursor at a specified horizontal location, then transitions to the interior state and repeatedly self-transitions back to that state, each time imaging a character and advancing the cursor, and finally terminates in the stop state, again at a location that has been specified in advance. Typically, the cursor locations specified for the start and stop states are the left- and right-most printable pixel locations in the image, respectively. A complete path through this subsource can be represented by a trellis diagram, wherein nodes represent horizontal pixel locations along the baseline, and edges represent the transitions that together make up the complete path. Each edge is labeled with a score that is the product of the prior probability of the transition and the likelihood of the corresponding imaged template in the spatial location that the edge spans. Finding the highest-score path can be accomplished by a straightforward application of dynamic programming.

2 Incorporating Linguistic Constraints in DID

As it is described above, DID makes no use of prior knowledge about which recognized transcriptions are more linguistically valid than others; it simply chooses the transcription corresponding to a path with highest posterior probability. It is desirable from the point of view of error rate to provide DID with a means of preferring linguistically more valid transcriptions over less valid ones. Several approaches are possible. Conceptually, the simplest is to modify the dynamic programming search to provide a multiplicity of high-probability paths rather than a single one, then select among them on the basis of linguistic validity [9]. Preliminary experiments following this approach reveal a potential for significant reduction in recognition error rates in some case, but only if the number of retained paths is very large (on the order of 500 per text line) [8].

In principle, a more direct way of incorporating linguistic constraints in DID is by modifying the Markov source model so that the states encapsulate linguistic context. We are primarily interested in *soft* linguistic constraints, which can be expressed probabilistically within this linguistically-expanded Markov source. The main obstacle to doing this in a direct way is that the requisite number of states grows exponentially in the length of the linguistic contexts being considered. In speech recognition, the potential exponential explosion of states (and hence candidate paths) is usually addressed by modifying the search algorithm to explore only a small fraction of candidate paths, namely those deemed most promising as the search unfolds. While this approach appears to be useful in practice in finding good transcriptions, it provides no guarantee that the path with highest probability will be actually be found [2, Chapter 6].

In this paper we consider an alternative search strategy that is guaranteed to find the most probable path, while in practice avoiding exponential growth in space and time complexity. Linguistic constraints are expressed by a *language model*, which measures linguistic validity by assigning conditional probabilities to characters in a sequence. The idea is to find a best candidate path using upper bounds on the language model probabilities, then to rescore this path using improved bounds or actual language model probabilities, repeating these two steps until the best candidate path found has been scored only with actual probabilities rather than upper bounds. The success of this approach in avoiding exponential growth relies on the empirical fact that the likelihoods (template matches against the image) play a much greater role in determining the best path than do the probabilistic linguistic constraints. In practice, the linguistic constraints exert only a "tiebreaking" influence.

2.1 Probabilistic Language Model

Linguistic validity can be measured by means of a probabilistic language model, which in its full generality is a probability distribution over all finite strings over a given alphabet. For use in document image decoding we restrict the language model to be factorable as a sequence of probability distributions over individual characters, each conditioned on a subset of preceding characters. Let the alphabet be $\mathcal{A}$, and let $v_1, \ldots, v_n$ denote a string with $v_i \in \mathcal{A}, i = 1, \ldots, n$. Let τ be a termination symbol, and let $\mathcal{A}' = \mathcal{A} \cup \{\tau\}$. We view strings as being formed by the following process. Characters are generated sequentially according to a sequence of conditional probability distributions

$$p_i(v_i | v_1, \ldots, v_{i-1}) = p_i(v_i | \phi_i(v_1, \ldots, v_{i-1})) \tag{1}$$

where $v_i \in \mathcal{A}'$, $v_1, \ldots, v_{i-1} \in \mathcal{A}$, $i = 1, 2, \ldots$, and the function $\phi_i(v_1, \ldots, v_{i-1})$ maps contexts into equivalence classes. The string terminates when the symbol τ is generated; in terms of the Markov source this corresponds to a transition into the stop state. This is essentially a tree source similar to the source described in [7], but differs from it in its use of a termination symbol and in the fact that it induces a valid probability distribution over all finite strings.

For simplicity in this paper, we remove the dependence of p in (1) on i, and restrict $\phi_i(v_1, \ldots, v_{i-1})$ to be of the form

$$\phi_i(v_1, \ldots, v_{i-1}) = \begin{cases} (v_1, \ldots, v_{i-1}) & \text{if } i \leq N \\ (v_{i-N+1}, \ldots, v_{i-1}) & \text{otherwise} \end{cases} \tag{2}$$

for a fixed small integer N. With these restrictions, (1) is referred to as a *character N-gram language model*, hereinafter referred to simply as an N-gram model:

$$p_i(v_i | v_1, \ldots, v_{i-1}) = p(v_i | v_{i-N_i+1}, \ldots, v_{i-1}) \tag{3}$$

where $N_i = \min\{i, N\}$. Although the search technique to be described herein remains practical using a class of language models more general than N-gram, N-grams are widely used and are fairly effective in capturing important statistical regularity in natural language strings, so that much of the potential improvement in recognition accuracy accruing from the use of arbitrarily complex language models is preserved under this restriction, and much is gained in the way of clarity of exposition.

For a fixed N-gram, we define a sequence of auxiliary functions $q_0, \ldots, q_{N_i-1}$ as

$$q_k(v_i | v_{i-k}, \ldots, v_{i-1}) = \max_{v_{i-N_i+1}, \ldots, v_{i-k-1}} p(v_i | v_{i-N_i+1}, \ldots, v_{i-1}) \tag{4}$$

which for each k provides an upper bound on the probability that can be assigned by the N-gram to v_i when immediately preceded by $(v_{i-k}, \ldots, v_{i-1})$. For example, q_0 specifies the maximum probability that can be assigned by the model to v_i in *any* context, while at the other extreme, q_{N-1} is simply another name for p. Note that for any fixed string section $(v_{i-N_i+1}, \ldots, v_i)$, q_k is nonincreasing in k.

3 Incorporating an N-gram Language Model in Best-Path Search

The posterior probability of a complete path through the Markov source can be expressed as the product of two factors: the likelihood of the path given the observed image, and the prior language-model probability assigned to the associated transcription. From the point of view of minimizing error rate on a given document image, we wish to find a maximum posterior probability (MAP) complete path [6] in a practicable manner. Inspired by a technique for reducing computations in standard DID [4], we propose finding a MAP path by iteratively refining candidate paths, each time using improved upper bounds on linguistically weighted edge scores. However, departing from [4], we apply the search within text lines, use a sequence of upper bound functions derived from a language model, and incur an expansion of state on each iteration. If the number of iterations necessary for convergence is small, as is expected when the images are relatively clean, then the expansion will not be prohibitive.

3.1 Iterated Complete-Path Search Algorithm using N-grams

We assume an N-gram language model with the functions p and $\{q_0, \ldots, q_{N_i-1}\}$ as defined above. Initially, a standard DID trellis is constructed as described in Section 1.1, except that the score on each edge is now the product of the template likelihood and the unconditional upper bound q_0 on the language model probability assigned to the corresponding character label. An initial best candidate path $\pi^{(0)}$ is found by dynamic programming in the usual way. The following two steps are then iterated:

1. For each *non-maximal* node along the current candidate best path $\pi^{(j)}$, a new node corresponding to the same spatial location is created, and a context label is associated with the new node corresponding to the transcription of the longest unambiguous path segment leading to it. A node is deemed *maximal* if its associated context label is of length $N_i - 1$.
2. A new candidate best path $\pi^{(j+1)}$ is found in the expanded graph by restricting the dynamic programming search to consider only those nodes in each spatial location that have the most specific context consistent with path history. Edge scores used in the search are the product of the template likelihood and highest-order upper bound q (or probability p, in the case of a maximal originating node) consistent with the originating node's context label.

The search terminates when all nodes along $\pi^{(j)}$ in step (1) are found to be maximal; such a path is provably MAP with respect to the language model and template likelihood scores.

4 Simulation on a Demonstration Problem

Application of the search technique described in Section 3 to text recognition is most easily understood and analyzed in terms of a one-dimensional demonstration problem. The essential ingredients are a message source, a signaling scheme that involves the concatenation of variable-width templates, and a degradation process whereby the concatenated templates are corrupted to yield the observed waveform to be decoded.

Table 1. American Morse Code.

Letter	Codeword	Letter	Codeword	Letter	Codeword	Letter	Codeword
A	·-	K	-·-	U	··-	5	·····
B	-···	L	·-··	V	···-	6	-····
C	-·-·	M	--	W	·--	7	--···
D	-··	N	-·	X	-··-	8	---··
E	·	O	---	Y	-·--	9	----·
F	··-·	P	·--·	Z	--··	0	-----
G	--·	Q	--·-	1	·----	.	·-·-·-
H	····	R	·-·	2	··---	,	--··--
I	··	S	···	3	···--	?	··--··
J	·---	T	-	4	····-		

4.1 Demonstration using American Morse Code

Specifically, we take the character templates to be discrete waveform segments obtained from the codewords of American Morse Code (Table 1), with each '·' replaced by the numeric sequence $(2, 3, 2, 1)$ and each '-' replaced by $(2, 3, 3, 2, 1)$. A letter space character is represented as $(1, 1, 1, 1, 1)$. When a string is "typeset" into a waveform, adjacent templates are separated by the spacer element (1). For example, the string `'THE'` maps to the waveform

$$(2,3,3,2,1,1,2,3,2,1,2,3,2,1,2,3,2,1,2,3,2,1,1,2,3,2,1)$$

After the waveform is created from the original text string, it is degraded by passing it through an additive white Gaussian noise channel. Specifically, pseudorandom noise is added to each element in the waveform, where the noise is independent and normally distributed with mean zero and variance σ^2.

Construction of the decoding trellis requires that the spatial locations corresponding to the start and stop states be specified. In actual practice these would be set to the left and right extremal positions, respectively, at which a template can be imaged. To facilitate synchronization, a special linguistically inert single-space template would be introduced into the set of character templates, and assigned a probability sufficiently small to discourage its gratuitous insertion. When constructing and matching linguistic conditioning contexts, all instances of this single-space template would be ignored. For simplicity of implementation, a single-space template is not used in the work reported here; instead, it is assumed that the true start and stop positions are known exactly in advance.

The body of text selected for preliminary experiments is an electronic version of Lewis Carroll's *Through the Looking Glass* [1], with all characters mapped to upper case, and only those symbols listed in Table 1 along with newlines and spaces retained. Blank lines are omitted, and the remaining 3,118 lines of text are divided into two equal-size portions: a training segment, consisting of the even-numbered lines, and a test portion, consisting of the odd-numbered lines.

4.2 Implementation of the N-gram

In the present study, an N-gram language model is implemented using a trie. Specifically, an N_i-long window is displaced sequentially from left to right along the training text, terminating alternately at every position in the text. At each position, characters in the window are scanned from the left, and the trie is descended (or grown, as necessary) recursively downwards, incrementing the count of each node encountered. The leaf counts are transformed into the requisite N-gram conditional probability estimates by adding a smoothing factor α to each, then normalizing.

Upper bounds are stored in the same trie. This is possible because of the manner in which training is done. Specifically, by sliding the context window exhaustively along the training text, it is ensured that for every suffix of a path from root to leaf, there exists a node in the trie that can be reached by descending that path-suffix from the root. This allows the upper bounds of order k to be recorded at the level-k nodes of the trie; in practice these are obtained by making a second pass through the training data using the probabilities estimated on the first pass.

Now consider applying the trained language model to score new data. While smoothing allows valid conditional probabilities and upper bounds to be assigned to characters in a given context that were missing in the training data, it does not help when the required context node itself has a count so low as to make the probability estimate unreliable, or worse, when the context node is missing from the trie entirely. In this not-so-uncommon situation, "backing off" is used: suffixes of the desired context of decreasing length are tried until the resulting context node has a count that is larger than a specified threshold value m. In such cases, the resulting context nodes are considered to be of maximal order when testing for termination of the iterated best-path search.

4.3 N-gram Parameters

The existence of a count-based backing-off strategy mitigates the data sparsity problem, and allows N to be larger than might otherwise be warranted [3]. Several combinations of values of N, α, and the count threshold m were tried in training the language model on the training data, and the resulting log-likelihood values on the test data noted. Suitable crossvalidated parameter values were thus found to be $N = 4$, $\alpha = 0.025$, and $m = 5$, resulting in an estimated coding cost on the test data of 2.24 bits per character.

4.4 Preliminary Results

Document Image Decoding was implemented for the Morse Code problem using data structures suitable for implementation of the search method described in Section 3. Errorful transcriptions were obtained for a variety of values of the channel noise intensity parameter σ, ranging from 0.05 to 0.50. The resulting transcription error rates for line-by-line decoding for the first fourteen lines of the test data, as measured by edit distance to the original text, is shown in Figure 1. For comparison, the error rates using the Viterbi algorithm without a context-dependent language model are also shown. From the graph it is evident that incorporating the language model via the proposed search algorithm significantly reduces error rates in this example. One of the characteristics of the pro-

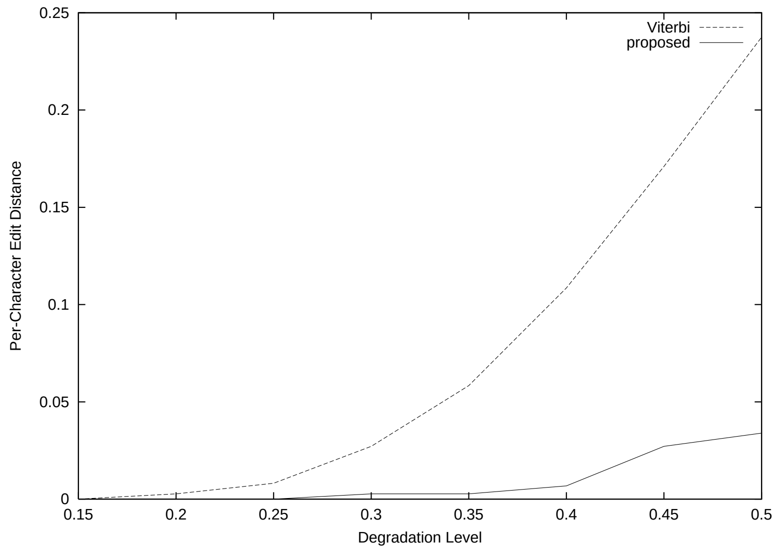

Fig. 1. Per-character transcription error rates versus channel noise intensity σ for the proposed algorithm, as measured by edit distance, using unit costs for insertions, deletions, and substitutions. Also shown is the corresponding curve for the stock Viterbi algorithm sans language model.

posed search algorithm is that its space- and time- complexities depend strongly on the degree to which the template likelihood scores dominate the overall path scores. In the low-noise case, the template likelihoods overwhelm the language model scores, so that the algorithm converges relatively quickly to a path found early on in the search, with little expansion of state beyond the starting trellis. At the other (high-noise) extreme, the templates provide very little information, and the decoder must essentially "hallucinate" a high-probability string of the required length using the language model alone. This latter task is essentially exponentially complex in the order of the language model, so we expect a computational explosion as the degradation level increases. We can see this effect in Figure 2, where the number of iterations required by the proposed algorithm is plotted against noise level. The corresponding space-complexity curve shown in Figure 3. Finally, in Figure 4 we present the cumulative raw path scores for the proposed algorithm, Viterbi sans language model, and ground truth. Paths found by the proposed algorithm consistently outscore Viterbi and match or

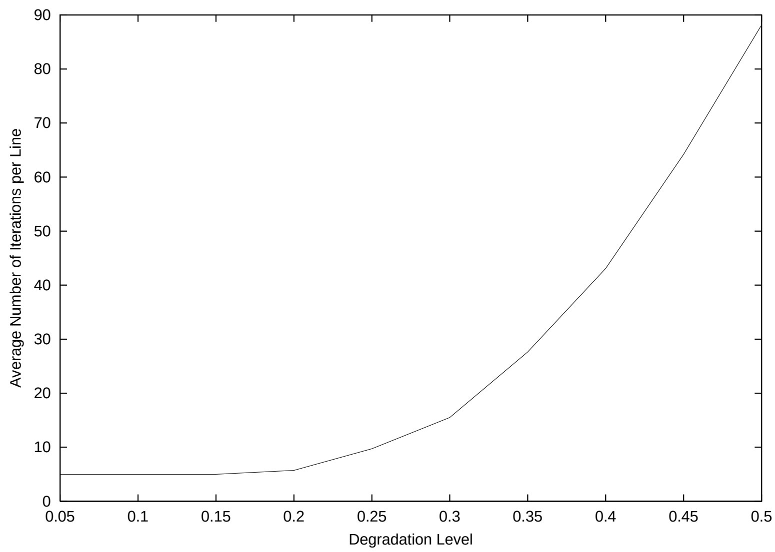

Fig. 2. Average per-Line number of iterations required by the proposed algorithm for convergence, versus channel noise. Based on the same simulation run as Figure 1.

outscore ground truth, which is expected as the algorithm is guaranteed to find a path with the highest score. When the components of the overall score were examined, it was found that the Viterbi algorithm generally results in higher likelihoods, while the proposed algorithm tends to yield higher language-model scores. Again, these findings are consistent with expectation, considering what each algorithm optimizes.

5 Conclusion

An iterative technique for incorporating a class of causal, sequentially predictive probabilistic language models into the document image decoding framework has been described in detail. Its two theoretical advantages over standard approaches are that is integrated directly into the best-path search algorithm, and it results in a truly optimal decoding with respect to the language and degradation models. Complexity is controlled by expanding state only on an as-needed basis, so that in the usual low-noise case where the template likelihoods dominate the path scores, only a modest expansion in space- and time-complexity occurs. However, in cases of moderate to severe degradation, the computational complexity of the proposed algorithm becomes prohibitive. Ultimately, the cause of this behavior is the algorithm's insistence on finding a "best" path rather than just a "good" path, the latter which may well be sufficient in practice.

Preliminary results on a demonstration problem involving recognition of a one-dimensional Morse Code signal indicate a potential for significant reduction

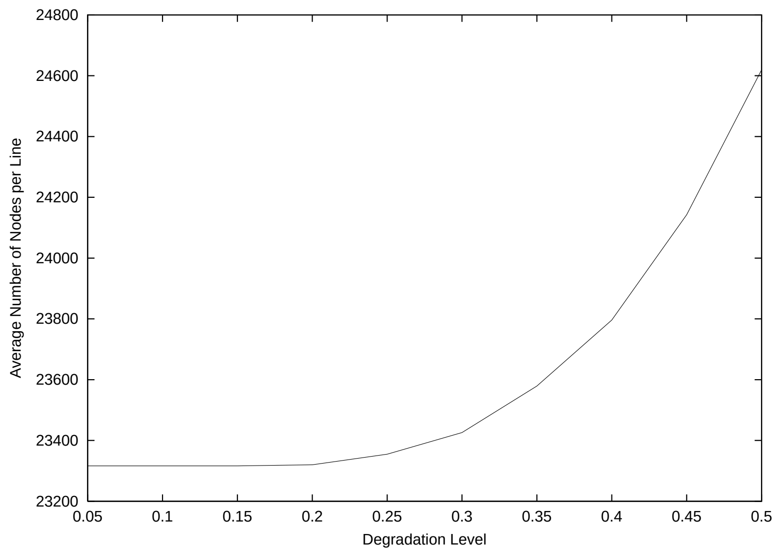

Fig. 3. Space-complexity (average per-Line number of graph nodes) required by the algorithm versus channel noise. The baseline corresponds to the size of the starting trellis.

in error rates by incorporating even a modest language model into DID using the proposed algorithm. The next step in exploring the nature and possible uses of this algorithm will be a more extensive simulation on actual document images, taking into account the synchronization issues alluded to in Section 4.1.

It is still an open question whether the proposed algorithm can offer clear practical advantages over alternative techniques that find only approximate best paths. More immediately clear is the usefulness of the proposed algorithm as a research tool. At moderate computational cost, it provides a means of finding an otherwise elusive true-best path when a non-trivial language model is in the system. This provides useful calibration information when evaluating the performance of alternative, perhaps approximate search strategies.

6 Acknowledgments

This work has benefitted from discussions with Henry Baird, Tom Breuel, Francine Chen, Gary Kopec, Jia Li, and Les Niles.

References

1. L. Carroll. *Through the Looking Glass.* Project Gutenberg, Milennium Fulcrum edition, 1991.

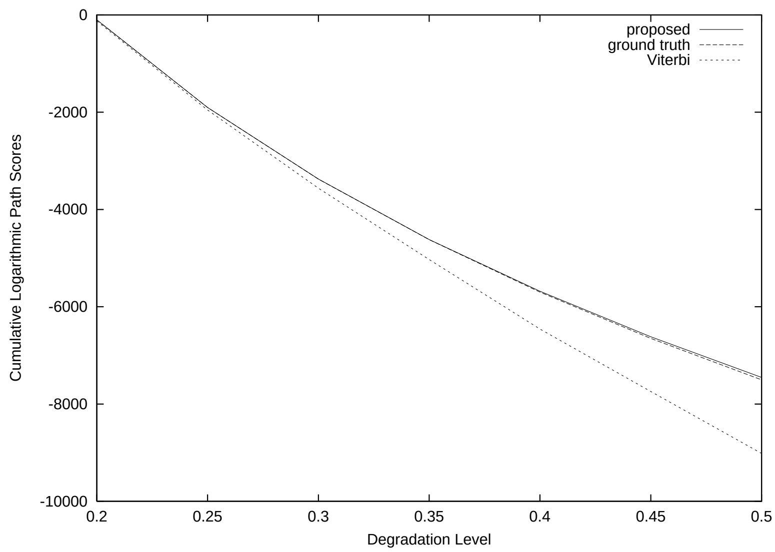

Fig. 4. Cumulative path scores for the proposed algorithm, the Viterbi algorithm sans language model, and ground truth.

2. F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, Massachusetts, 1997.
3. F. Jelinek, R. L. Mercer, and S. Roukos. Principles of lexical language modeling for speech recognition. In S. Furui and M. M. Sondhi, editors, *Advances in Speech Signal Processing*, chapter 21, pages 651–699. Marcel Dekker, New York, 1992.
4. A. C. Kam and G. E. Kopec. Document image decoding by heuristic search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):945–950, September 1996.
5. G. E. Kopec. Multilevel character templates for document image decoding. In *Proceedings of the IS&T/SPIE 1997 Intl. Symposium on Electronic Imaging: Science & Technology*, San Jose, February 1997.
6. G. E. Kopec and P. A. Chou. Document image decoding using Markov source models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):602–617, June 1994.
7. J. J. Rissanen and G. G. Langdon. Universal modeling and coding. *IEEE Trans. Inform. Theory*, IT-27:12–23, 1981.
8. M. R. Said. Unpublished work. Xerox Palo Alto Research Center, CA, August 1997.
9. R. Schwartz and S. Austin. A comparison of several approximate algorithms for finding multiple (n-best) sentence hypotheses. In *ICASSP 91: 1991 International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 701–704, Toronto, Ont., Canada, May 1991. IEEE.