

N-Gram Language Models for Document Image Decoding

Gary E. Kopec^a, Maya R. Said^b, Kris Popat^a

^aXerox Palo Alto Research Center, Palo Alto, CA 94304.

^bMassachusetts Institute of Technology, Cambridge, MA 02139.

ABSTRACT

This paper explores the problem of incorporating linguistic constraints into document image decoding, a communication theory approach to document recognition. Probabilistic character n-grams ($n=2-5$) are used in a two-pass strategy where the decoder first uses a very weak language model to generate a lattice of candidate output strings. These are then re-scored in the second pass using the full language model. Experimental results based on both synthesized and scanned data show that this approach is capable of improving the error rate by a factor of two to ten depending on the quality of the data and the details of the language model used.

Keywords: document image decoding, Markov models, document recognition, character recognition.

1. INTRODUCTION

Document image decoding (DID) is an approach to document image recognition in which the processes of document image creation, transmission and interpretation are modeled as a formal communications system consisting of an image source, a channel and a receiver. Applying the DID paradigm to a specific recognition problem involves developing probabilistic models for the source and channel. One of the components of a text source model is a description of the allowed character sequences and their probabilities. The text models originally used in DID placed no constraints on the allowed character sequences and took all characters to be equally probable. This places the entire burden of recognition on the character shape models. If the images are relatively clean and the character models are accurate the unconstrained text model can support very accurate recognition. For reliable performance under less ideal conditions, however, a more sophisticated language model is advantageous.

This paper describes our earliest attempts to use probabilistic character n-gram ($n = 2-5$) models in text image decoding. In principle, an n-gram language model can be incorporated within a Markov image source and used directly in a maximum a posteriori (MAP) decoder. However, the number of model states can be very large (e.g. over 1,000,000 for $n=5$) which makes such a direct approach computationally challenging. Here, we follow a two-pass strategy, similar to the two-pass n-best methods common in speech recognition,^{1,2} in which the first-pass decoder uses a very weak language model (e.g. $n=1$) to generate a lattice of candidate output strings. These are then re-scored in the second pass using the full language model.

This paper is organized as follows: Section 2 presents a review of document image decoding. Language models are described in Section 3 including the character n-gram model used in this paper. The problem of incorporating n-gram language models into DID is taken up in Section 4 where different strategies are reviewed and the two-pass strategy is described. Finally, experimental results are presented in Section 5.

Gary Kopec initiated and supervised this work in 1997; he passed away in December 1998. Any errors or deficiencies in this work are solely the responsibility of the other authors.

This work was performed while Maya Said was at the Xerox Palo Alto Research Center.

Author email addresses: mayasaid@mit.edu, popat@parc.xerox.com

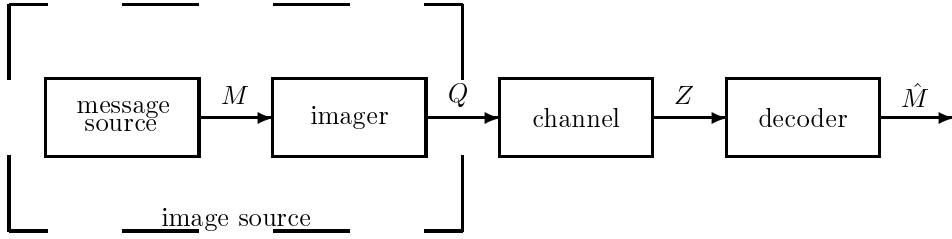


Figure 1: Communication Theory View of Document Recognition³

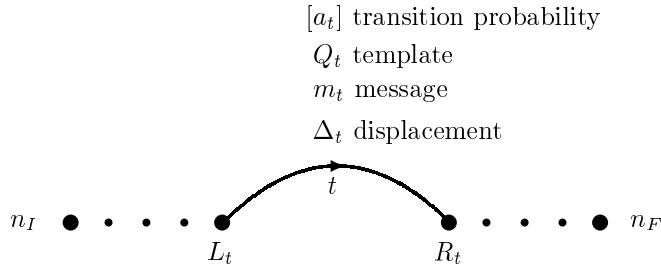


Figure 2: Markov Source Model for Image Generation³

2. REVIEW OF THE DOCUMENT IMAGE DECODING FRAMEWORK

Document Image Decoding (DID) was first introduced and formulated by Kopec and Chou in 1994.³ It defines a methodology based on a communication theory view of document recognition. In this section, we briefly review the DID formulation and algorithm implementation, and in so doing, motivate the introduction of language models. The reader is referred to Kopec and Chou³ for a more detailed treatment.

Figure 1 illustrates the communication theory view of document recognition. Document creation, transmission, and interpretation are modeled as the transmitter (image source), channel, and receiver (decoder) in a communication system. Specifically, the image is modeled as being first generated using a probabilistic *image source* which selects a finite string from a set of candidate strings according to a prior probability distribution $P(M)$. The string is then mapped to an ideal image Q by the *imager*, which “prints” ideal templates corresponding to the selected string onto the image plane at appropriate locations. We assume that the templates are bilevel and therefore Q is a binary image, i.e. $Q = \{q_i | i \in \Omega\}$ where $q_i = \{0, 1\}$ are the pixel values with 1 denoting the foreground color and 0 the background color, and Ω is the image plane which can represent a single document page or multiple pages arranged in a scroll. More specifically, the image source can be modeled as a Markov source as shown in Figure 2 with a starting state, n_I , and a final trapping state, n_F . Each transition t connects a pair of states L_t and R_t and is associated with four attributes (Q_t, m_t, a_t, Δ_t) where Q_t is the template, m_t is the message string (typically a single character), a_t is the transition probability and Δ_t is the vector displacement of t . A complete path through the source starts at n_I and ends at n_F and defines a composite message, an image, and a probability associated with this path. The composite message is the concatenation of the individual message strings associated with each transition. The image is composed of the union of the templates properly positioned according to their displacement vectors. The probability of a path, $P(\pi)$, is the product of the transition probabilities. Note that in order to get the probability of the composite message or image, one needs to sum the probabilities of all complete paths that lead the same composite message and image. The ideal image Q is then converted into an observed image $Z = \{z_i | i \in \Omega\}$ by the *channel* which models distortions introduced due to scanning, printing, or photocopying. A simple, yet powerful, model for the channel is the asymmetric bit-flip noise model shown in Figure 3 where pixels of values 0 and 1 have respectively probability α_0 and α_1 of being transmitted correctly. The noise parameters are assumed to be constant

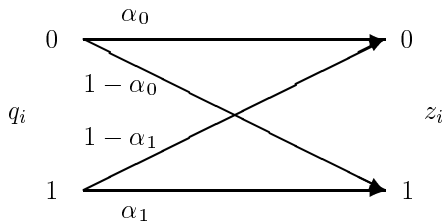


Figure 3: Asymmetric bit-flip Noise Model.

over the image. An important generalization of this idea involves *multilevel* templates, wherein the transition probabilities are allowed to depend not only on the value of each pixel but also on its “class” or “level” within the ideal template.⁴ The purposes of the present study in language model integration are sufficiently served by assuming the simple case of bilevel templates, which corresponds directly to Figure 3.

Finally, the *decoder* produces an estimate, \hat{M} , of the original message M based on the received image Z . The minimum probability of error decoder is the one that chooses \hat{M} according to the maximum a posteriori (MAP) criterion: $P(\hat{M}) = \max_M P(M|Z)$. A direct implementation of the MAP rule would involve a summation over all complete paths that generate the same composite message. However, one can simplify this implementation by using the *Viterbi approximation*, which chooses the path with the greatest probability for a given composite message, i.e. the *Viterbi decoder* chooses the complete path that maximizes:

$$\mathcal{L}(\pi, Z) = \log(P(\pi)) + \mathcal{L}(Z|Q_\pi) \quad (1)$$

where $\mathcal{L}(Z|Q_\pi)$ is the log normalized probability that the entire observed image Z results from transmitting the shifted templates Q_t through the channel. Based on experience in speech recognition, the Viterbi approximation is a good approximation to the MAP decoder and can be implemented efficiently using dynamic programming.

In the original DID formulation, no constraints were placed on the allowed character sequences and all characters were considered equally likely. In the context of Figure 2, this corresponds to having all transition probabilities associated with message strings equal or equivalently $P(\pi)$ is constant. As a result, the most likely path through the network given an observed image is determined by the template match scores rather than by the path prior probabilities, i.e. the entire burden of recognition is placed on the character shape models. This can lead to accurate recognition if the images are relatively clean and the character models are accurate; however under less ideal conditions, a more accurate language model becomes essential. While it would be straightforward to assign unequal probabilities to the transitions in the network, experience has shown that doing so is insufficient to significantly impact recognition accuracy. A more powerful language model structure is required, one that moves beyond the marginal, unconditional statistics of individual characters and takes into account inter-character dependence. The specifics of a suitable language model are taken up in the following section.

3. LANGUAGE MODEL PROPERTIES

A language model is a mathematical formulation of linguistic constraints. It assigns a probability value to every string over some specified set of language building blocks, such as words, phonemes, characters, or possibly combinations of these. Language models based on words and phonemes have been extensively used in speech recognition to guide the search among various text hypotheses, and have proved to greatly improve recognition accuracy.⁵ However, such models are not directly applicable to character recognition since they do not support punctuation, upper and lower-case letters, acronyms, numbers, etc. We therefore explore purely character-based language models, where the meaning of *character* is expanded to include punctuation and white spaces.

The character-based language model probability P_l for a string $[\gamma_1, \dots, \gamma_m]$ can be generally factored using the chain rule for probabilities:

$$P_l = \prod_{i=1}^m P(\gamma_i | \gamma_1, \dots, \gamma_{i-1}) \quad (2)$$

where $P(\gamma_i | \gamma_1, \dots, \gamma_{i-1})$ is the probability of obtaining a character γ_i immediately following the string $[\gamma_1 \dots \gamma_{i-1}]$. A convenient way to rewrite Equation (2) is in terms of the log probabilities:

$$\log(P_l) = \sum_{i=1}^m \log [P(\gamma_i | \gamma_1, \dots, \gamma_{i-1})] \quad (3)$$

As the length of the string being conditioned on within each conditional probability in Equation (2) increases, obtaining a good estimate for that probability becomes increasingly hard. It is reasonable to assume, however, that statistical dependence between characters weakens with distance within the string, so that the probability of observing a character γ_i given that $n - 1$ preceding characters were observed can be a good approximation to the probability of observing γ_i given the entire past history of characters observed. That is, we may assume that

$$P(\gamma_i | \gamma_{i-n}, \dots, \gamma_{i-1}) = P(\gamma_i | \gamma_1, \dots, \gamma_{i-1}) \quad (4)$$

for some suitable value n . We therefore chose to restrict the language models to fixed length sequences of characters called *character n-grams* where n is the length of the sequence. For every n-gram in the vocabulary, the language model therefore assigns a probability value. We next discuss how to estimate these probabilities.

3.1. N-Gram Probabilities

A simple approach to estimating n-gram probabilities is to use the method of maximum likelihood within each context, which amounts to strict relative frequency. Specifically, given a training corpus such as an encyclopedia, we can estimate the n-gram probabilities as follows:

$$P(\gamma_i | \gamma_{i-n}, \dots, \gamma_{i-1}) = \frac{C(\gamma_{i-n}, \dots, \gamma_{i-1}, \gamma_i)}{C(\gamma_{i-n}, \dots, \gamma_{i-1})} \quad (5)$$

where the output of the function $C(x, y, \dots, z)$ is the number of character strings ‘ $xy\dots z$ ’ contained in the training corpus. A problem however arises if the value of the probabilities in Equation (5) is zero, i.e. if the n-gram appears in the text on which the recognizer is run but is not contained in the training corpus. We will refer to such n-grams as *missing* or *non-occurring* n-grams. Assigning the maximum-likelihood probability of zero to such n-grams would lead to an error when computing the log probability, and more importantly would not allow for new words to be incorporated into the model. If one has good confidence in the training corpus, one could still chose to reject missing n-grams by assigning them an infinitesimal probability, for instance $\log P(\gamma_i | \gamma_{i-n}, \dots, \gamma_{i-1}) = -10,000$. This method will be referred to as *rescore-1* or *R1* throughout this paper. A more flexible yet simple language model, on the other hand, updates the n-gram probability by, for example, assuming the missing n-grams occurred exactly once in the training corpus, i.e. by assigning it the probability:

$$P(\gamma_i | \gamma_{i-n}, \dots, \gamma_{i-1}) = \frac{1}{C(\gamma_{i-n}, \dots, \gamma_{i-1})} \quad (6)$$

Strictly speaking, the probabilities of all other n-grams with the same $n - 1$ preceding characters should be updated due to the addition of the new n-grams (i.e. $C(\gamma_{i-n}, \dots, \gamma_{i-1})$ should be incremented by one). However, in practice this change is very small and the updated probabilities are very close to the original one. Note that strict normalization is not required for the role played by $P(\pi)$ in Equation (1). Moreover, since the role of the language model here is only in re-scoring hypotheses that have already been identified as good on the basis of image match, $P(\pi)$ need not be a valid probability distribution at all, as long as it provides reasonable differentiation among competing hypotheses. Therefore, we chose not to increment those counts that were nonzero to begin with. This method will be referred to as *rescore-2* or *R2*.

A remaining issue is that the conditioning string itself might be absent from the training corpus, i.e., the denominator in Equation (5) might be zero. In such cases we simply assign the infinitesimal value $P(\gamma_i|\gamma_{i-n}, \dots, \gamma_{i-1}) = 10^{-6}$, or equivalently $\log(P(\gamma_i|\gamma_{i-n}, \dots, \gamma_{i-1})) = -6$. Again, we note that the fact that the resulting quantity is not normalized to yield a strictly valid probability distribution is not a concern here; it is only important that the language model score provide a reasonable relative ranking among hypotheses identified in the first pass.

More sophisticated methods of handling both of the above types of data sparsity — the training-set absence of observed characters within a context, and the absence of the context itself — have been reported in the literature.¹ The methods described here were deemed adequate for the purposes of exploring the potential value of language model integration, especially given the modest values of n considered, the availability of large training corpora, and the “post-processing” role played by the language model.

4. INCORPORATING N-GRAM LANGUAGE MODELS INTO DID

While the language model probabilities are simple to evaluate along a particular path (i.e., for a particular string), searching for a best path is not as straightforward as in the case of traditional DID because of the introduction of statistical interdependencies among characters. This section discusses how the search can be practically undertaken, that is, how the language model can be incorporated into DID.

4.1. Infeasibility of Direct Incorporation

In terms of the network shown in Figure 2, the effect of the language model is to make the transition probabilities assigned to each edge depend on the recent path history leading to that edge, violating the Markov assumption. To restore it, we can in principal expand the state space to include not only position within the image, but also linguistic context. A moment’s reflection reveals that this will not be practical even for modest n , since the state would have to expand by a factor that is exponential in n . For example, for a 2-gram model with 100 different characters in the font, the number of states would increase by a factor of 10,000. As a result, direct incorporation of the language model into the Viterbi algorithm is computationally prohibitive. In Section 4.3, we present an efficient algorithm for incorporating the language model into DID based on a two-pass strategy. While the approach described there represents our earliest attempt at language modeling and is the main subject of this paper, it should be noted that two other strategies have been explored during the three-year interim between the completion and publication of the present work. We therefore review these other approaches briefly in the following section.

4.2. Stack and Iterated Complete Path Algorithms

In the absence of a language model, the search for the most probable path can be represented as a search for a highest-weight path in a trellis graph that represents the template match scores at every position in the image; the search can be accomplished effectively in this case using the Viterbi algorithm. At the other extreme, with an unconstrained language model of the sort given in Equation (2), the graph to be searched over would be a tree rather than a trellis. In the case of an n -gram language model with a restricted conditioning history, the tree merges back onto itself into a complex trellis structure, the exhaustive search of which is impractical in general.

One approximate search technique that is popular in speech recognition and which was recently explored in the DID context⁶ involves partially exploring this tree-like graph using a best-first search strategy. This technique, known as the *Stack* algorithm, involves growing a subtree until a transition into n_F is encountered. Leaves of the subtree are maintained in a priority queue according to an estimate of how “promising” each one is, given its fully evaluated partial path score, together with a prediction of the score of the path remaining to n_F . When a node is popped off the queue, its children are immediately scored and placed on the queue. The accuracy and computational complexity of this approach depend critically on how well the relative “promise” of nodes at different depths are estimated.

Another approach, dubbed the *iterative complete path algorithm*, also explores only a small fraction of the immense tree-like graph, but differs from the Stack algorithm in that it returns a provably best path.⁷ It works

by growing the original trellis on an as-needed basis, expanding linguistic state iteratively only along paths that could not be ruled out on an earlier iteration on the basis of insufficient template match score under a best-case language-model score scenario. Here, the language model components of the edge scores are optimistic upper bounds that depend, as the graph is grown, on fewer characters than the full language model. A path that survives is re-scored by incrementing the order of the upper-bound language model used to score its edges and correspondingly growing the graph to accommodate the increased linguistic context. When a path is found whose edges have been re-scored to the point where all of them have full language model scores, it can be concluded that it is a true best path, since it has beaten all of the optimistically scored paths either present or implicit in the expanded trellis.

Both of these techniques for language model integration are relatively complex; the first has been found thus far to be somewhat fragile in the DID context; and the second is not guaranteed always to be computationally tractable (though in practice it has often been found to be). It is therefore desirable to explore a simpler, computationally strictly bounded alternative technique, to serve as a baseline and to further build understanding of the problem of language model integration. The two-pass approach described in the following section meets this need.

4.3. Two-Pass Strategy

The two-pass strategy is based on a separation of the recognition process based on character shape (the Viterbi algorithm) from the one based on language constraints. The decoder is first run with a weak language model to get the k -best sentences with their corresponding scores $S_d = \mathcal{L}(\pi, Z)$. A more sophisticated language model is then applied to the k -best sentences and a language score $S_l = \log(P_l)$ is computed. Finally, the sentences are re-sorted according to their total score S_T where S_T is defined as a weighted sum of the decoder score and the language score:

$$S_T = S_d + \lambda S_l \quad (7)$$

where λ is a scaling factor needed to adjust the weighting attributed to the language model. The best sentence is the one with the best total score.

Clearly this strategy relies on the existence of a good linguistic solution in the k -best sentences generated by the original decoder, i.e. the language model is only used to fine-tune the estimate out of the Viterbi decoder. Theoretically, for a large enough k , the correct solution will always be present among the k -best sentences. However, obtaining the k -best sentences out of the Viterbi algorithm is only computationally feasible for low enough k . It turns out as we shall see in the next section that even for low k values, incorporating a language model leads to an amelioration of the error by more than 50%

5. EXPERIMENTAL RESULTS

The effect of incorporating n-gram language models into DID using the two-pass strategy was investigated using two sets of experiments. In a first set, we used synthesized data where an electronic collection of Shakespeare sonnets was artificially distorted using the bit-flip model illustrated in Figure 3. The second set was performed using the English journal subset of the University of Washington UW-II database⁸ which contains scanned images from technical journal articles.

5.1. Synthesized Data

An electronic collection of Shakespeare sonnets with a total of 59,761 characters was first distorted using the bit-flip channel model with parameters ranging from $[\alpha_0 = 0.9, \alpha_1 = 0.9]$ to $[\alpha_0 = 0.7, \alpha_1 = 0.5]$. The 20 best sentences with their associated scores were then generated using the original Viterbi algorithm with both matched channel parameters and unmatched parameters modeling perfect and imperfect knowledge of the channel at the decoder. A language score was then computed for each sentence using an n-gram (n=2-5) language model. Two sets of language models were used: in the first set, the n-gram probabilities were computed using the entire Shakespeare sonnets collection as a training set (a total of 4,822,274 characters) while the Grolier encyclopedia was used for the second set (53,034,883 characters). The Shakespeare collection included 2,134 distinct 2-grams, 17,716 3-grams, 81,289 4-grams, and 246,454 5-grams while the Grolier collection included

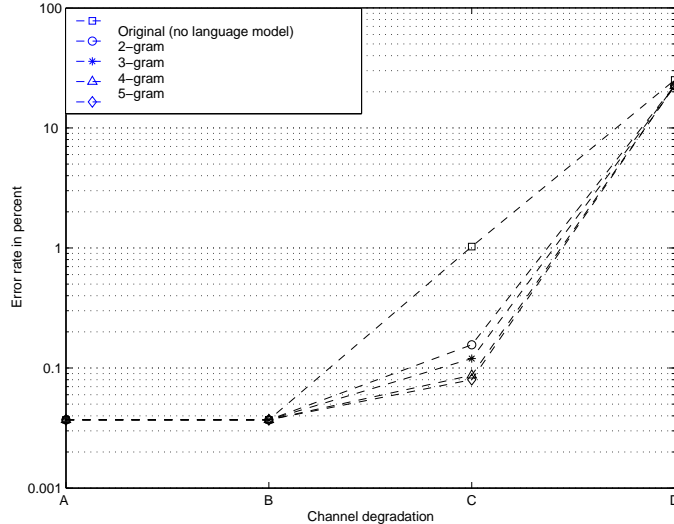


Figure 4. Character Recognition Error Rate as a Function of Channel Degradation for the Shakespeare Data. A, B, C, and D correspond to channel parameters: $[\alpha_0 = 0.9, \alpha_1 = 0.9]$, $[\alpha_0 = 0.9, \alpha_1 = 0.8]$, $[\alpha_0 = 0.8, \alpha_1 = 0.6]$, and $[\alpha_0 = 0.7, \alpha_1 = 0.5]$ respectively. The same channel parameters were used for the decoding. The Shakespeare database was used as a training set for the language model. The missing n-grams were rejected (*rescore-1*).

4,582 2-grams, 55,286 3-grams, 351,650 4-grams, and 1,260,849 5-grams. In the process of computing the language model score, missing n-grams were either rejected (*rescore-1* or *R1*) or incorporated (*rescore-2* or *R2*) as described in Section 3.1. Finally, a total score was computed for each sentence according to Equation (7) where the scaling factor λ was varied until the best recognition was achieved. The 20 sentences were resorted according to the new score and the best one was identified.

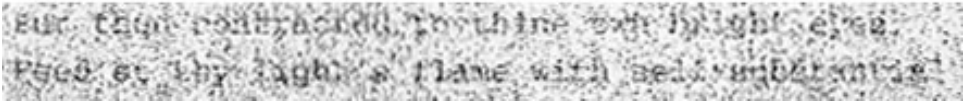
Figure 4 shows the recognition error rate as a function of the noise in the image for different n-gram orders. The language model fails to improve the recognition performance at very low (A and B in the figure) and very high (D) noise levels. This behavior can be explained by looking at the output of the Viterbi algorithm under various noise regimes. At low noise levels, the 20 best sentences obtained from the Viterbi algorithm correspond to the same message string, therefore resorting them cannot lead to better performance. At very high noise levels, on the other hand, the 20 best sentences correspond to different message strings that do not include a good solution, therefore the language model is not able to improve recognition drastically. The error rate however is still improved by a factor of two. The major contribution of the language model is observed at intermediate noise levels (C in the figure) where recognition is improved by more than an order of magnitude. An example of a degraded verse and its decoded versions using the original Viterbi algorithm as well as the language model is shown in Figure 5

The effect of n-gram order on error rate is shown in Figure 6 where the character recognition error rates for intermediate noise levels (channel parameters $[\alpha_0 = 0.8, \alpha_1 = 0.6]$) are plotted as a function of n-gram order for the two re-scoring methods (*rescore-1* and *rescore-2*) for the Shakespeare-trained language model, and for *rescore-2* for the Grolier-trained model. (Data for the remaining combination, Grolier/*rescore-1*, is no longer available.) Clearly, using a 2-gram language model greatly improves recognition over the original Viterbi algorithm with no language model. The two *rescore-2* curves track each other closely, with the better-matched model performing slightly better throughout. For n larger than 2, the Shakespeare-trained language model using *rescore-1* results in significantly better error rates than the *rescore-2* curves. Overfitting may be part of the explanation here; the missing curve might have helped assess the extent to which it is. However, overfitting cannot be the whole explanation, since a similar effect appears in Figure 7, as will be discussed in greater detail shortly.

Table 1 summarizes the character recognition results for the Viterbi algorithm with no language model, the

(A)
 But thou contracted to thine own bright eyes,
 Feed'st thy light's flame with self-substantial

(B)



(C)
 But thou contracted to thine own'bright eyes,
 Feed'st thy light's'flame with self-substantial

(D)
 But thou contracted to thine own bright eyes,
 Feed'st thy light's flame with self-substantial

Figure 5. Illustration of the Decoding Results for the Shakespeare Sonnets. (A) corresponds to the original uncorrupted image, (B) is the corrupted image using channel parameters $[\alpha_0 = 0.8, \alpha_1 = 0.6]$. The decoding result using the Viterbi algorithm with no language model is shown in (C). Finally, (D) corresponds to the re-scored result using a 2-gram language model trained on the Grolier dataset and designed to incorporate missing n-grams (*rescore-2*). The full text corresponding to this example included a recognition error rate of 1.026% with the original Viterbi algorithm which was reduced to 0.161% using language model rescoring.

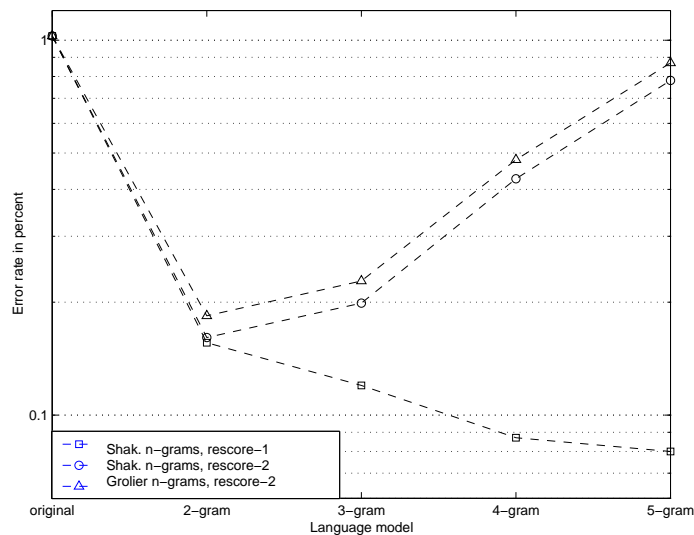


Figure 6. Character Recognition Error Rate as a Function of n-gram Order for Channel Parameters $[\alpha_0 = 0.8, \alpha_1 = 0.6]$ with Matched Decoding. *original* refers to the original Viterbi algorithm with no language model. *Shak* and *Grolier* refer to, respectively, the Shakespeare and Grolier datasets used to obtain the n-gram probabilities. In *rescore-1* missing n-grams were rejected while *rescore-2* incorporated missing n-grams.

2-gram and 3-gram language models trained on Shakespeare data using the *rescore-1* and *rescore-2* algorithms. The results in Table 1 suggest that the language model leads to a ten-fold decrease in error rate primarily due to a decrease in substitution and insertion errors. Interestingly, the language model leads to almost a doubling of deletion errors. This is due to the fact that a white space is treated as a character when computing the n-gram probabilities. The probabilities of n-grams containing spaces is therefore much higher than other n-grams since

Method	Subs.	Dels.	Ins.	Total	Distinct	% Error
Viterbi algorithm without a language model	563	29	21	613	33	1.026
Shak n-grams with <i>rescore-1</i>						
2-gram	39	54	0	93	19	0.156
3-gram	18	54	0	72	17	0.120
Shak n-grams with <i>rescore-2</i>						
2-gram	41	55	0	96	21	0.161
3-gram	63	56	0	119	24	0.199

Table 1. Character Recognition Results for the Shakespeare Data with Channel Parameters [$\alpha_0 = 0.8, \alpha_1 = 0.6$]. *Subs* = substitutions, *Dels* = deletions, *Ins* = insertions, *Total* = total number of errors, *Distinct* = total number of distinct errors, *% Error* = percent error rate.

n-grams with white spaces occur very frequently in any training set (at the end and beginning of every word.)

5.2. Scanned Data

In a second set of experiments, the UW-II English journal database was used as a testing set. The database contains 622 bilevel scanned images of 311 pages from 30 technical journal articles. For each page, two images are provided which correspond to images scanned from first and second generation photocopies. Article “A” (i.e. W0A and W1A) in the database was used to run the experiments because it uses a sans-serif font which makes ‘l’ and ‘I’ essentially indistinguishable. As a result, a decoder that exclusively relies on character shape models is expected to generate a large number of substitution errors. The article included 43,696 characters and the original Viterbi algorithm was first used to obtain the 10, 100, and 500 best sentences with their corresponding scores. The *rescore-1* (*R1*) and *rescore-2* (*R2*) algorithms were then used to re-score these sentences using two language models trained on the ground-truth provided in the UW-II collection and the Grolier encyclopedia, respectively. The sentences were then sorted according to their new score and the best one was selected.

Figure 7 summarizes the performance of the different order n-gram models using the 10, 100, and 500 best sentences (panels (a), (b), and (c)). The results suggest that incorporating a language model *always* leads to an improvement of the recognition error ranging from 56% to 85%. As in the synthesized data case, the error rate is sensitive to the re-scoring method for high order n-grams where the *rescore-2* (*R2*) method (incorporating missing n-grams) performs worse than the *rescore-1* (or *R1*) method. Since the *R1* curves in Figure 7 reflect training and testing on completely different datasets, it is clear in this case that the good performance of *R1* is not due to overfitting.

One can gain additional insight as to why *R1* outperforms *R2* for large *n* in correctly rejecting strings with missing n-grams by noting that while the individual probabilities of missing n-grams in the *R2* method is still low (of the order of the probability of the least occurring n-gram), the total language score of a sentence containing a missing n-gram can potentially be much higher than one not containing a missing n-gram but is otherwise identical to the first one. This is due to the fact that the number of n-grams that a given character influences increases with the n-gram order. For example, in a 4-gram model, the character ‘d’ in the text string ‘*abcdefg*’ appears in four 4-grams, namely ‘*abcd*’, ‘*bcde*’, ‘*cdef*’, and ‘*defg*’. While ‘*abcd*’ may be a missing n-gram, the probability of ‘*bcde*’, ‘*cdef*’, and ‘*defg*’ may be much higher than their corresponding n-grams in the text string ‘*abchefg*’ where now ‘*abch*’ is an occurring n-gram. In this scenario, the text string ‘*abcdefg*’ will have a much higher probability than the text string ‘*abchefg*’ if scored with the *rescore-2* method while the *rescore-1* method will favor the string ‘*abchefg*’. As the n-gram order increases, the performance of the *rescore-2* method will therefore degrade as shown in Figure 7. Of course the same argument works against *R1* if the n-grams were missing in the training data but present in the correct decoding. In such cases, *R2* would be expected to assign a higher score to the correct decoding, since *R1* would essentially rule it out. Evidently from

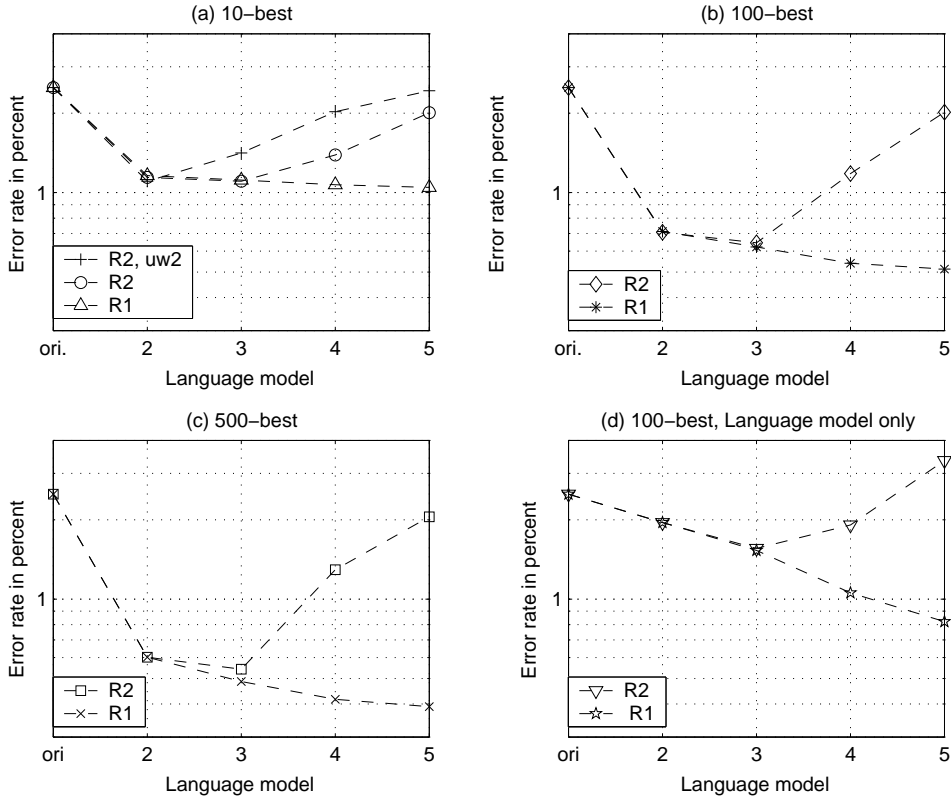


Figure 7. Character Recognition Error Rate as a Function of n-gram Order for the UW-II data. In panel (a), the 10-best sentences were used for rescoring. The 100-best and 500-best sentences were used in panels (b) and (c) respectively. Panel (d) shows the result of rescoring the 100-best sentences using the language model only (i.e. setting S_d to zero in Equation (7)). The probabilities for the language model were computed using the Grolier encyclopedia dataset for all curves except the one labeled *uw2* where the UW-II dataset was used to infer the probabilities. The x -axis labels are as follows: *ori*: Original Viterbi algorithm with no language model, 2, 3, 4, 5 = 2-gram, 3-gram, 4-gram, and 5-gram language models respectively. Finally, *R1* and *R2* correspond to the *rescore-1* and *rescore-2* algorithms respectively.

Figure 7, n-grams in the Grolier encyclopedia are effectively a superset of the n-grams present in the correct decoding of the UW-II article, so that *R1* dominates for all n .

The contribution of the language model to the error rate was also investigated by rescoring the 100-best sentences according to the language model score exclusively, i.e. S_d in Equation (7) was set to zero and $S_T = S_l$ was used to re-sort the sentences, the result is shown in panel (d) of Figure 7. While the error rate slightly improves for the *rescore-1* method, it deteriorates for the 5-gram model using the *rescore-2* method. These results strongly suggest that a combination of *both* character shape model and language model recognition is needed to get the best performance. An example of the scanned data and the recognition results is shown in Figure 8. As expected, the Viterbi decoder includes many ($1 \rightarrow I$) substitutions which are corrected by applying the language model. Finally, Table 2 summarizes the recognition error details for the scanned data. Again as is the case for the synthesized data, the language model improvement is essentially through a reduction of substitution and insertion errors while deletions are greatly increased.

6. SUMMARY

A two-pass strategy for incorporating language models into document image decoding has been described. The Viterbi decoder using a very weak language model ($n=1$) is first used to generate a lattice of candidate

(A)
Road tracking in remotely sensed imagery has often been equated with linear feature extraction. The rationale was that finding

(B)
 Road tracking in remotely sensed imagery has often been equated with linear feature extraction. The rationale was that finding

(C)
 Road tracking in remotely sensed imagery has often been equated with linear feature extraction. The rationale was that finding

Figure 8. Illustration of the Decoding Results for the UW-II Data. (A) corresponds to the scanned image, the decoding result using the Viterbi algorithm with no language model is shown in (B). Finally, (C) corresponds to the rescored result using a 5-gram language model applied to the 500-best sentences. The language model used was trained on the Grolier dataset and designed to reject non-occurring n-grams. The full text corresponding to this example generated an error rate of 2.499% with the original Viterbi algorithm and 0.378% with the language model.

Method	Subs.	Dels.	Ins.	Total	Distinct	% Error
Viterbi algorithm without a language model	911	1	180	1092	53	2.499
100-best, Grolier n-grams with <i>R1</i>						
2-gram	205	23	83	311	48	0.712
3-gram	162	57	62	281	49	0.643
4-gram	158	32	52	242	48	0.554
5-gram	151	24	49	224	46	0.513
100-best, using language model only, Grolier n-grams with <i>R1</i>						
2-gram	222	570	62	854	59	1.954
3-gram	207	400	61	668	56	1.529
4-gram	215	190	57	462	58	1.057
5-gram	193	120	46	359	57	0.822
100-best, using language model only, Grolier n-grams with <i>R2</i>						
2-gram	214	576	62	852	61	1.950
3-gram	220	421	40	681	53	1.558
4-gram	458	328	49	835	68	1.911
5-gram	887	472	107	1466	70	3.355

Table 2. Character Recognition Results for the UW-II Data. *R1* and *R2* correspond to rejecting and incorporating missing n-grams, respectively. *Subs* = substitutions, *Dels* = deletions, *Ins* = insertions, *Total* = total number of errors, *Distinct* = total number of distinct errors, *% Error* = percent error rate.

output strings. These strings are then re-scored using probabilistic character n-gram (n=2-5) language models. Preliminary results using both synthesized and scanned data suggest a factor of two to ten reduction in character error rate depending on the noise level in the data, the number of candidate strings, the n-gram order, the training corpus, and the way missing n-grams are treated. In general, the choice of training corpus did not greatly affect recognition rate while rejecting, as opposed to incorporating, missing n-grams proved to be a better strategy for high order n-gram language models. When the rejecting missing n-grams strategy was used, the choice of n-gram order did not significantly affect the error reduction rate.

Relative to other approaches to language model integration, *k*-best rescoring offers several advantages. The first is robustness against producing poor decodings: only hypotheses that are deemed sufficiently good in

the first pass are candidates for output in the second pass. A related benefit is that the language model requirements are relaxed; in particular, it is not necessary to maintain a strictly valid probability distribution. Another advantage is conceptual simplicity. A potential disadvantage is the possibility of missing the best hypothesis if it happens not to be among those selected in the first pass. This is particularly a danger when the degradation level is fairly high. More work needs to be done to understand the computational and accuracy tradeoffs for the various approaches to language model integration under various noise regimes. This work has made it clear, however, that the potential improvement in decoding accuracy by incorporating a language modeling into DID is significant. The recent explorations into the use of the Stack and ICP algorithms were motivated largely by the (then unpublished) findings of improved accuracy presented here.

7. ACKNOWLEDGMENTS

The authors wish to thank Dan Bloomberg and Les Niles for many stimulating discussions throughout the course of this work.

REFERENCES

1. F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Massachusetts, 1997.
2. R. Schwartz and Y.-L. Chow, "The n-best algorithm: An efficient and exact procedure for finding the n most likely sentence hypotheses," in *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech and Signal Processing*, **1**, pp. 81–84, (Albuquerque, NM, USA), 1990.
3. G. E. Kopec and P. A. Chou, "Document image decoding using Markov source models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**, pp. 602–617, 1994.
4. 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.
5. F. Jelinek, R. L. Mercer, and S. Roukos, "Principles of lexical language modeling for speech recognition," in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi, eds., ch. 21, pp. 651–699, Marcel Dekker, New York, 1992.
6. K. Popat, D. Greene, J. Romberg, and D. S. Bloomberg, "Adding linguistic constraints to document image decoding: Comparing the iterated complete path and stack algorithms," in *Proceedings of IS&T/SPIE Electronic Imaging 2001: Document Recognition and Retrieval VIII*, January 2001. To appear.
7. K. Popat, D. Bloomberg, and D. Greene, "Adding linguistic constraints to document image decoding," in *Proceedings of the 4th international workshop on document analysis systems*, International Association of Pattern Recognition, December 2000.
8. I. T. Phillips and R. M. Haralick, "UW English/Japanese document image database II: A database of document images for OCR research." CD-ROM, available from the Intelligent Systems Laboratory, Dept. of Electrical Engineering, University of Washington, Seattle, WA 98195, att.: Dr. Robert M. Haralick, March 1995.