

## **Use of Language Models in Handwriting Recognition**

Rohini Srihar, Shravya Shetty and Sargur Srihari

TR-06-07  
June 2007



Center of Excellence for Document Analysis and Recognition (CEDAR)  
520 Lee Entrance, Suite 202  
Amherst, New York 14228

# Use of Language Models in Handwriting Recognition

Rohini K. Srihari, Shravya Shetty , Sargur N. Srihari

{rohini,sshetty,srihari}@cedar.buffalo.edu

Center of Excellence for Document Analysis and Recognition (CEDAR)

Department of Computer Science and Engineering

UB Commons, 520 Lee Entrance- Suite 202

State University of New York at Buffalo

Buffalo, NY 14228-2567

## Abstract

Language models have been extensively used in natural language applications such as speech recognition, part-of-speech tagging, information extraction, etc. To a lesser extent the value of language models in text recognition has also been proved, e.g., recognition of poor quality printed text and the recognition of extended handwriting. This survey describes how linguistic context, particularly probabilistic language models, are used in the recognition of handwritten text. The survey begins with two handwriting recognition techniques, segmentation-free and segmentation-based, are integrated with language models in the recognition process. Next, language models at the word level in the post processing step to improve the recognition results and at the character level for handwriting recognition

and correction of recognition results are described. Finally, syntax based techniques like lexical analysis using collocations, syntactic (n-gram) analysis using part-of-speech (POS) tags, and a hybrid syntactic technique comprised of both a statistical and an analytical component are described. Language modeling has been found to be very helpful for all natural language applications. They have been seen to improve the performance of these application by 25-50% when the text used in training is representative of that for which the model is intended.

## I. INTRODUCTION

Language models based on bigram and trigram probabilities have been extensively used in natural language applications such as speech recognition, part-of-speech tagging, information extraction, etc. To a lesser extent the value of language models in text recognition has also been proved, e.g., recognition of poor quality printed text and the recognition of extended handwriting. Such applications have been largely limited to the processing of the Latin alphabet, i.e., the script in which English is written. Here we describe possible approaches to use probabilistic language models in the recognition of handwritten text.

The use of context is important in handwriting recognition, [47] describes the use of context in several applications of pattern recognition. Figure 1 demonstrates the importance of context in recognizing handwritten text. The first example shows the ambiguity in identifying the middle character as ‘h’ or ‘a’. Here, the context of the previous and the next character helps in identifying the character as ‘h’ when viewed horizontally and ‘a’ when viewed vertically. Similarly in the second example the context of characters is used to identify the middle character as ‘B’ and the context of other numbers identifies it as the number ‘13’. The ambiguity when viewing a character by itself is resolved when viewed in context.

The central component of a handwritten text recognizer is a word recognizer (WR). The input to a WR is a word signal and a lexicon. Its output consists of an ordered list of the best  $n$  words in the lexicon which match the word signal. Due to wide variability in writing, WRs often do not return the correct word as the top choice. It is clear that the key to improving raw WR output is to utilize contextual knowledge that accompanies the word signal. An important form of contextual knowledge is linguistic information.

As an example, consider Fig. 2 which shows the digitized image of the sentence “my alarm

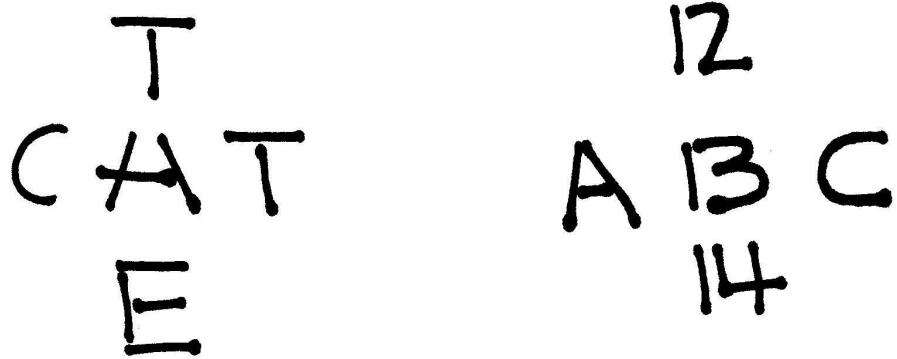


Fig. 1. Use of Context in identifying characters

clock did not wake me up this morning” along with the output of the WR. If we restrict the recognizer to return only the top choice for each input word, the sentence will be erroneously read as “my alarm code soil rout wake me up thai moving”. Although the correct words can be found in the top few choices, it requires the use of contextual constraints in order to override the (sometimes erroneous) top choice. Word recognizers have a tendency to misread short (less than or equal to 3 characters) words more frequently than longer words. Furthermore, short words tend to be pronouns, prepositions and determiners causing frequent word confusion between very different syntactic categories (e.g., as, an). The WR problem is further compounded by the possible absence of the correct word in the top  $n$  choices.

Word recognition models at both the character-level (to model words) and at the word-level (to model sentences) are described. Segmentation free and segmentation based word recognition techniques using language models are described. We discuss the use of language models at all stages of the recognition process. This includes the use of word and character ngram models in the recognition phase, the judicious use of lexicons in the holistic filtering stage, the use of dictionary statistics in the word recognition phase, and finally higher-level syntactic and

|           |              |  |   |   |             |
|-----------|--------------|--|---|---|-------------|
| <i>my</i> | <i>alarm</i> | <i>clock</i>                                 | <i>did</i>  | <i>not</i>                                | <i>wake</i> |
| my        | alarm        | code<br>circle<br>shute<br><b>clock</b>      | soil<br>raid<br>risk<br>visit<br>mail                   | rout<br>hot<br>list<br><i>not</i><br>most | not         |
| me        | up           | thai<br>taxis<br>trie<br>tier<br><b>this</b> | moving<br>having<br><b>morning</b><br>running<br>loving |   |             |
| me        | up           | thai<br>taxis<br>trie<br>tier<br><b>this</b> | moving<br>having<br><b>morning</b><br>running<br>loving |   |             |

Fig. 2. Isolated Word Recognition Output. Correct words are shown in bold; italicized lists indicate that correct word is not among top choices

semantic post-processing of the WR output. We also describe syntax based techniques and how feedback from syntactic and semantic processing can be employed in the word recognition process; all these techniques are designed to improve the WR rate. Some of these techniques have been discussed in [50],[48],[5], [40]. recognition results are created by specific integration of a language model in the hidden Markov model based recognition system. [50] and [5] discusses the effect of ngram language models on the performance of a hidden Markov model (HMM) based offline handwriting recognition system, [48] combines Hidden markov Models and Statistical Language Models for word recognition, [40] discusses statistical methods of applying linguistic constraints to the output of an WR on input consisting of sentences/phrases.

#### A. Relationship to Machine-print Recognition, Speech Recognition

There is considerable overlap between the techniques called for in handwriting recognition and those employed in machine-print and speech recognition. In fact, the linguistic techniques used in improving machine-print recognition are a subset of those required for handwriting recognition. The language used in everyday speech is similar to that used in handwriting, namely informal and frequently ungrammatical. For this reason, the higher-level language modeling described

will find applications in speech recognition as well.

Specifically, the areas of common interest include lexicon design (both general and domain-specific), and the use of Markov models of syntax in order to constrain the number of possible choices at every word position in a given sentence/phrase [17], [12], [23]. However, there are some fundamental differences in the three areas which justifies the need for developing new techniques particular to the handwriting domain.

The word recognition output in the case of machine-print is significantly better than that of handwritten word recognizers. First, the confidences output by the word recognizer are reliable enough to be used as a source of information in further processing. Secondly, it is possible to select a threshold  $t$  such that the average word neighbourhood size is reasonable, while ensuring that the correct word is almost always present. In the case of handwriting, confidences are not reliable; in many cases, only the relative rank of a word is taken into consideration when evaluating choices. Secondly, the correct word is often missing from the word neighbourhood even if a large number of choices (50) are considered. Thus, a significant effort is required to overcome the effect of missing words.

Both machine-print and speech recognition systems make use of context-free grammars for higher-level language modelling [32]. In the case of machine-print, the fact that printed documents represent edited English, combined with the wide availability of electronic corpora (representing several genres of texts) permit the development and use of formal grammars. On the other hand, handwritten text exhibits informal (unedited) use of language and is more difficult to classify into genres. Informal refers to the following properties: relatively short sentences or phrases, few sentences per document/message, frequent use of first-person pronouns (*I, me, my*), liberal use of abbreviations (*asap*), frequently ungrammatical, etc. Furthermore, due to

the difficulty in doing so, there has been no attempt to compile large corpora of handwritten text. In light of this, constructing formal grammars for handwritten text is extremely difficult at best and is not the approach taken in this research. Although speech recognition also suffers from the problems associated with informal language, the existence of numerous corpora, and domain restriction (e.g., travel reservation system) make the use of formal grammars a more viable option. Since speech recognizers are often used as interfaces to computer systems, they frequently involve an understanding component. In many instances, it is possible to obtain the gist of a command without having to transcribe every word correctly. The speech domain analogy to handwriting recognition is the dictation task. It is interesting to note that very little work has taken place in recognizing unrestricted, dictation.

### *B. Organization of Paper*

The remainder of the paper is organized as follows. Section II introduces language models for text recognitions, section III describes two handwriting recogniton techniques - segmentation free and segmentation based recogniton and the integration of language models in the recognition process. Section IV describes the use of language models at the word level in the post processing step to improve the recognition results. Section V describes language models at the character level for handwriting recognition and correction of recognition results. Section VI discusses syntax based techniques like lexical analysis using collocations, syntactic (n-gram) analysis using part-of-speech (POS) tags, and a hybrid syntactic technique comprised of both a statistical and an analytical component. Section VIII is a brief description of handling scripts other than the Latin script. Section IX provides a concluding discussion.

## II. STATISTICAL LANGUAGE MODELS

Statistical language models have been extensively used to improve the performance of natural language applications as they capture the regularities in a language. They have been used in applications such as speech recognition, machine translation, information extraction and spelling correction. Their value has been shown in optical character recognition (OCR) [20] and in the recognition of handwriting [39], [38], [40]. In general, the more powerful the language model is, the more computationally intensive it will be.

### *A. Language Models for text recognition*

The use of a language model for text recognition can be stated as follows. Given a text image, e.g., a scanned image of a sequence of printed or handwritten words, the most probable word sequence that produced the image can be stated as

$$\arg \max_{\text{word sequences}} P(\text{word sequence} | \text{text image}) \quad (1)$$

In principle, this optimization can be performed using either a generative model or a discriminative model [6]. Language models are typically used together with the generative approach. This can be seen by expanding equation 1 by using the Bayes rule as follows:

$$\begin{aligned} \arg \max_{\text{words}} P(\text{word sequence} | \text{text image}) &\propto \\ P(\text{text image} | \text{word sequence}) \times P(\text{word sequence}) \end{aligned} \quad (2)$$

The product on the right-hand side of the proportionality in equation 2 is just the joint probability of the word sequence and the word image. The first term in the product can be modeled generatively, e.g. it can be the product of the probabilities of segmented images given the corresponding tokens.

The second term of the product in equation 2 can be computed using a language model. Each word in the sequence depends on the entire history of words seen before it. Thus the probability of observing a sequence of  $n$  words is

$$\begin{aligned} P(\text{word sequence}_1^n) &= P(\text{word}_1) \times P(\text{word}_2|\text{word}_1) \times P(\text{word}_3|\text{word}_2, \text{word}_1) \\ &\quad \times P(\text{word}_4|\text{word}_3, \text{word}_2, \text{word}_1) \dots P(\text{word}_n|\text{word}_{n-1}) \end{aligned} \quad (3)$$

It is difficult to compute the probability of observing a word given all the words occurring before it. In an  $n$ -gram model every word in the sequence is assumed to depend only on the previous  $n - 1$  words. Equation 3 can be approximated using a  $n$ -gram model, equation 4 shows such an approximation using the bigram model. The most commonly used  $n$ -grams are unigram, bigram and trigram models.

$$\begin{aligned} P(\text{word sequence}_1^n) &= P(\text{word}_1) \times P(\text{word}_2|\text{word}_1) \times P(\text{word}_3|\text{word}_2) \\ &\quad \times P(\text{word}_4|\text{word}_3) \dots P(\text{word}_n|\text{word}_{n-1}) \end{aligned} \quad (4)$$

It is crucial that the text used in training is representative of that for which the model is intended.

### *B. Limitations*

The  $n$ -gram model can be learnt using the raw counts of the  $n$ -grams occurring in a training corpus. But a model learnt using the raw counts has certain limitations like that of sequences not occurring in the training sequence have a zero probability and many  $n$ -grams occur too few or too many times due to the nature of the corpus and hence skew the model prediction. Several improvements have been suggested to overcome these limitations like smoothing, caching, clustering, neural methods and sentence mixture models. In smoothing the probability mass estimated over N-grams appearing more than n times is redistributed across all possible sequences

of  $N$  words. In such a way a model trained over a certain corpus can be used for any other text, but  $N$ -grams that are not possible from a linguistic point of view will also have non-zero probability.

### C. Perplexity

The quality of a language model as applied to a given text can be measured in terms of the corresponding *perplexity* of the text. Intuitively, perplexity measures the average number of successor words which can be predicted for each word in the text (based on a language model). As an example, if the language model utilizes only first order transitions between successive words (bigrams), the perplexity  $P$  of a text with  $T$  words is:

$$P = \exp\left(-\frac{1}{T} \sum_{t=1}^T \ln(p(w_t | w_{t-1}))\right)$$

where  $p(w_t | w_{t-1})$  is the probability to get the word  $w_t$  given the word  $w_{t-1}$ . In the worst case, the perplexity is the same as the number of words in the text. Using higher values of  $n$  will lower the perplexity, however, estimating the conditional probabilities  $P(w_k | w_{k-1})$  becomes a problem. [9] describes a language model as a computational mechanism for obtaining these conditional probabilities.

## III. HANDWRITING RECOGNITION TECHNIQUES

Handwritten text recognitions involves three stages a)Preprocessing b)Recognition c)Post-processing. The preprocessing steps involve noise removal, binarization, segmentation. Figure 3 shows the different preprocessing steps described in [43]. Handwritten word recognition can be done using segmentation free or segmentation based methods. In the segmentation based methods every line of text to be recognized is further segmented into words and recognition

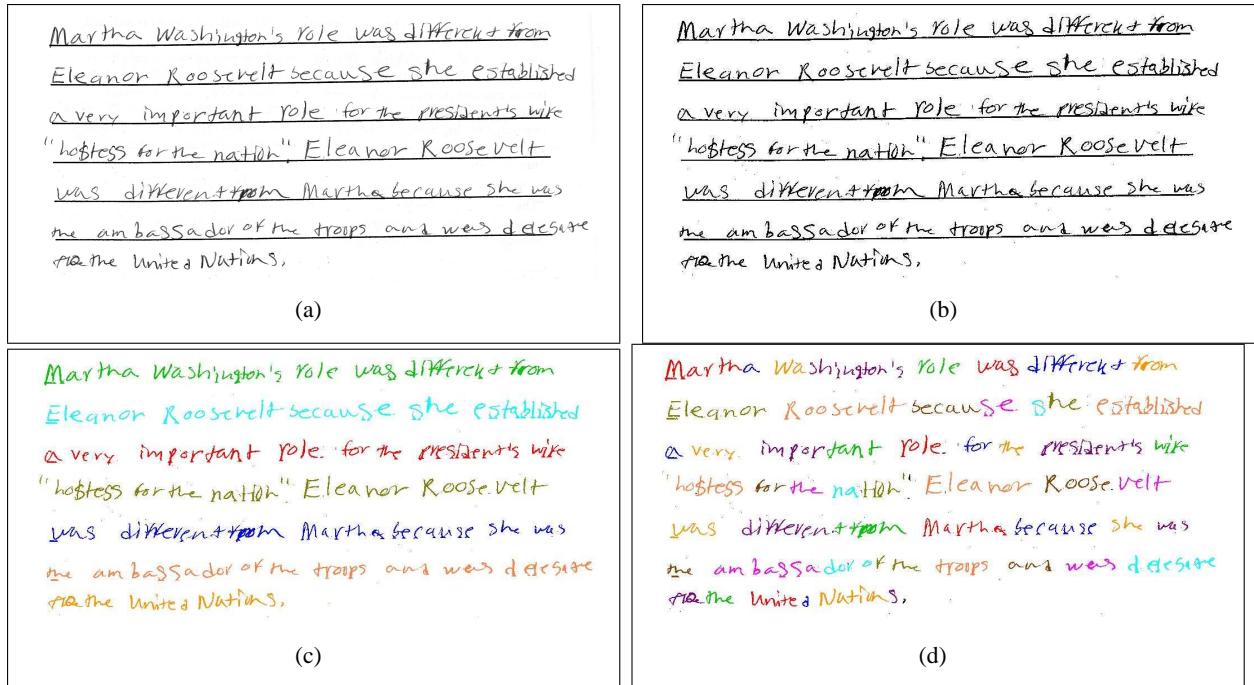


Fig. 3. Preprocessing steps: (a) Original document, (b) Binarization, (c) Line Segmentation and (d) Word Segmentation

is performed at the word level. Figure 3(d) shows the segmentation of a document into words. In segmentation based methods typically the recognition is done for every word image and language models are used as a post processing technique to improve the recognition results. These language models can be syntax based or semantics based. Due to the possible errors in segmentation several segmentation hypotheses might be considered. In case of the segmentation free methods the handwritten lines of text are not segmented into words. Here, language models are integrated in the recognition process rather than in the post processing step.

The word recognition itself may be at the word level or the character level. In case of word level recognition a holistic recognition of a word image is attempted. In case of character level recognition the individual characters are recognized. Here, the n-gram statistics at the character level can be incorporated into the model to aid recognition.

### A. Segmentation based approaches

Figure 4 illustrates a computational model for a segmentation based handwritten sentence/phrase recognizer which reflects the possible use of language models at every stage of the recognition process. The word separation step is where lines in the input image are segmented into words and punctuation thus producing a stream of word/punctuation images for the next stage (the size of the buffer is much smaller in online recognition (often, only one word)). The word buffer is fed into a lexicon filter whose purpose is to use intrinsic image features (such as word length, presence of ascenders/descenders) in reducing the lexicon. Word recognition consists of “segmentation” of the word image followed by character recognition in order to generate candidate (possibly illegal) words.

String matching techniques are subsequently employed to produce a *neighborhood* of legal words (based on the reduced lexicon). It is important to note that due to problems in separating words correctly, the number of output word neighborhoods may be different from the number of input words. It is expected that the ensuing syntactic and semantic processing will detect and correct some of these errors.

The next stage, namely semantic filtering, uses collocational information to modify word neighbourhoods. In this phase, new words (not present in the word neighborhoods) can be proposed. The input to this stage consists of a sentence/phrase buffer containing the word neighborhoods for each word. Syntactic processing uses statistical Markov models of syntax in order to reduce the word neighborhoods. Since these are primarily local syntactic models (using a context of 2 or 3 words at most), further syntactic and semantic processing is required in order to select the best sentence choice(s). The final stage of processing allows feedback to the word recognizer as indicated by the dashed lines in Figure 4. There are two types of feedback

provided: (i) feedback information to the WR post-processor in terms of eliminating syntactic categories from contention, or (ii) feedback to word recognition e.g., if syntactic analysis has determined that a particular token must be alphabetic only (as opposed to mixed alphanumeric), this information could be incorporated in a second “reading” of the word image.

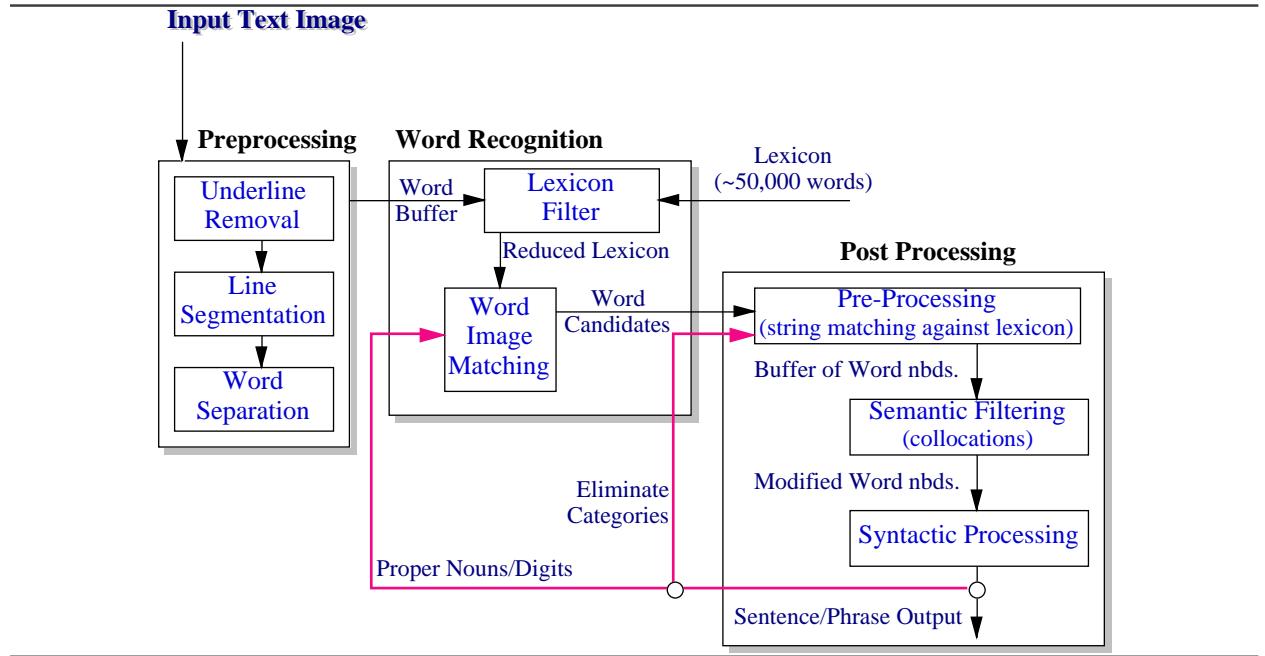


Fig. 4. Functional architecture of a handwritten text recognition system

*1) Recognition:* In the segmentation based methods, recognition is attempted at the word level for each segmented word image. The  $N$  best list of word recognition results are used to find the most likely sequence of words. The method of word recognition adopted in [25], [45] was one that combined the results of two processes, one an analytic recognizer and the other a holistic recognizer. The former is based on character shapes and the latter uses the global shapes of words. The results of the two methods are combined using a weighting scheme. The analytic recognizer relies on segmenting words into characters and recognizes them using a dynamic programming algorithm. The holistic recognizer uses known prototypes of handwritten words and matches them against segmented words in the scanned image. An optimal weighting

scheme for the two distance values is found using a simple neuron classifier with two inputs corresponding to the two distance values and a single output specifying the probability that the input corresponds to the distances of a non-matching word.

[13] uses a character model of word recognition (CMWR) which performs flexible segmentation, character recognition and graph matching in order to arrive at the best word candidates. [10] uses a word model of word recognition (WMWR) based on a Hidden Markov Model [10]. The final word output is computed by a third module which combines the ranking of the outputs of these two modules.

Several HMM based methods were developed for recognition in segmentation based recognition systems. HMMs cope with dynamic property and noise. They can be used to avoid character segmentation. Letter sub-HMMs were also used to handle over and under-segmentation cases. Transitions between every state pair is allowed, a state can transition to the same letter sub HMM or it can transition to a different letter model. Linguistic statistics related to letters and pairs of letters occurrence frequency is derived from the lexicon.

The objective of recognition in a bayesian framework is to find  $\arg \max_W P(W|x) = P(W) \prod P(w_i|x_i)$ .  $P(W)$  can be obtained from the language model. The language model can be phrase based or word based. The phrase based models are suitable when the data set consists of short phrases. A dictionary of phrases can be learnt from a large corpus and the frequencies of the phrases can be used to approximate  $P(W)$ . But the phrase based model is not very scalable and is not suitable for most applications. The word based models are effected by errors in segmentation and by the presence of characters like ‘/’ , ‘:’, etc. [8] makes a comparison of the performance of phrase and word based models. It was found that phrase based models performed poorly when many of the phrases in the test set were not part of the language model.

The performance of the unigram word model was poor due to the large entropy and perplexity of the word model as compared to the phrase model. The performance was seen to improve when the output of recognizers based on the two language models was combined.

### *B. Segmentation free approach*

The approach of using language models after the recognition to correct the recognition results has the limitation of separating the recognition of handwritten data from the application of language modeling. Most segmentation free approaches overcome this limitation by integrating the language modeling into the handwriting recognition process.

An approach based on HMMs and N-grams was used in [34]. This approach has the important advantage of avoiding the segmentation (which is a byproduct of the recognition). In the recognition the language model constrains the possible next words. The HMMs incorporate the language model in the transition probabilities between states. During the training of the HMM, the language model plays no role. It is used in the decoding phase when recognizing a test sentence.

[48] describes a segmentation free approach which combines Hidden markov Models and Statistical Language Models for word recognition. Recognition at the word level is done using continuous density HMMs and the language models capture the transition between words. The search space is a network of word HMMs and the search is performed at the state and word levels. At each observation vector the state-word pair belonging to the globally most likely path is found. Two dynamic programming recurrence equations are used to search through the state network. As with other word level language model methods the recognition process becomes computationally intensive when as the length of the word sequence increases.

The N-grams word models have an important advantage: they can be easily included in a

decoding technique that integrates recognition and language modeling. The integration of other language models into decoding algorithms has been tried in speech recognition and it leads to poor performance improvements with respect to the increase of complexity it requires [22]. [33] makes a comparison of different language models in the recognition of unconstrained handwritten sentences. The language models they compare include a ‘simple sentence model’ which assumes a uniform distribution over all words and that they are independent of each other, a ‘unigram sentence model’ the uniform probability is replaced by the occurrence probability of the word and a ‘bigram sentence model’ which assumes that every word in the sentence depends on its previous word.

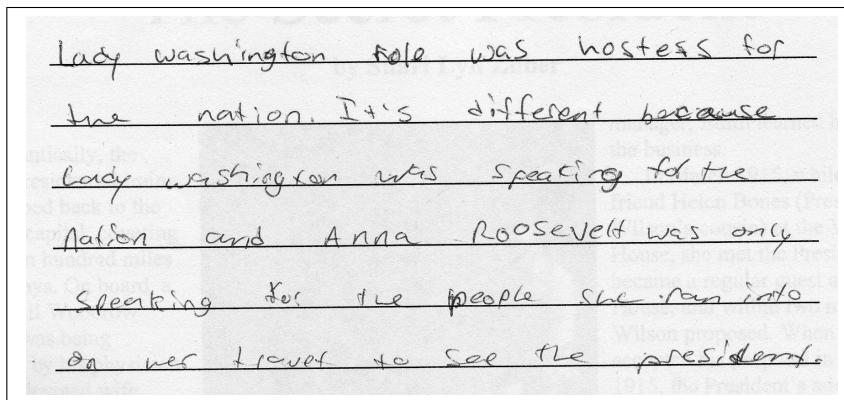


Fig. 5. Original document

#### IV. LANGUAGE MODELS FOR POST-PROCESSING

##### A. String Matching

Since the candidate words produced by the word recognition phase may not be legal words, a string matching procedure between candidate words and the lexicon is required. [27] contains a detailed discussion of isolated-word error correction techniques. String matching algorithms rank the lexicon according to a distance metric between a candidate word and each word in the lexicon. The Levenshtein distance is a commonly used distance metric. The ranked lexicon

| ORIGINAL TEXT  | WORD RECOGNITION  | LANGUAGE MODELING   | Four Essays |
|--|---|---|-------------|
| Lady Washington role was hostess for the nation. It's different because Lady Washington was speaking for the nation and Anna Roosevelt was only speaking <b>for the people</b> she ran into <b>on wer travet</b> to see the president. | lady washingtons role was hostess for the nation first to different because lady washingtons was speeches for for martha and taylor roosevelt was only meetings <b>for did people</b> first vote polio <b>on her because</b> to see the president | lady washingtons role was hostess for the nation but is different because george washingtons was different for the nation and eleanor roosevelt was only everything <b>for the people</b> first ladies late <b>on her travel</b> to see the president | 204         |
|  |   |   | 124         |
|  |   |   | 145         |
|  |   |   | 24          |

Fig. 6. Sample result showing improvement in recognition results

consists of sets of words  $n_1, n_2, \dots, n_t$  such that all the words in a set  $n_i$  have the same value for the distance metric. From the ranked lexicon, word recognition systems typically select a *neighborhood*  $N$  of words which (hopefully) contains the intended word.

In [36] a variation of the Levenshtein distance metric which incorporates edit costs in ranking the lexicon is used. Edit costs in handwriting are based on the probabilities of substitutions, merges, and splits occurring; this is in contrast to edit costs encountered in typographical systems, namely, substitution, insertion and deletion. The dynamic programming algorithm described in [49] was adapted [35] to take into account the likelihood of errors due to splitting (e.g.,  $u$  being recognized as  $ii$ ) and merging errors (e.g.,  $cl$  being recognized as  $d$ ). Furthermore, the errors were quantized into one of three categories and weights have been determined for each type of error.

The above method only takes into account channel information. String matching can be improved by incorporating dictionary statistics (i.e., the frequency of letter transitions, words, in the training data). [18], [16] discuss some techniques (and possible pitfalls) for doing the above.

However, there are several more techniques specific to the handwriting domain which can be brought to bear. As an example, if the candidate word consists of one or two characters, all words in the lexicon of length 1 or 2 are output. This is to compensate for frequent errors in recognizing very short words.

### B. Contextual Word Correction

The errors in recognition using the above mentioned segmentation based methods can be reduced using linguistic context in the form of transitional probabilities between words, between parts-of-speech tags, etc. The performance of a WR system can be improved by incorporating statistical information at the word sequence level. The performance improvement derives from selection of lower-rank words from the WR output when the surrounding context indicates such selection makes the entire sentence more probable. Given a set of output words  $\bar{X}$  which emanate from a noisy channel (such as an WR),  $N$ -gram word models [2], [1] seek to determine the string of words  $\bar{W}$  which most probably gave rise to it. This amounts to finding the string  $\bar{W}$  for which the *a posteriori* probability

$$P(\bar{W} | \bar{X}) = \frac{P(\bar{W}) * P(\bar{X} | \bar{W})}{P(\bar{X})}$$

is maximum, where  $P(\bar{X} | \bar{W})$  is the probability of observing  $\bar{X}$  when  $\bar{W}$  is the true word sequence,  $P(\bar{W})$  is the *a priori* probability of  $\bar{W}$  and  $P(\bar{X})$  is the probability of string  $\bar{X}$ . The values for each of the  $P(X_i | W_i)$  are known as the channel (or confusion) probabilities and can be estimated empirically. If we assume that words are generated by an  $n$ th order Markov source, then the *a priori* probability  $P(\bar{W})$  can be estimated as

$$P(\bar{W}) = P(W_{m+1} | W_{m+1-n}) \dots P(W_1 | W_0) * P(W_0)$$

where  $P(W_n | W_{k-n} \dots W_{k-1})$  is called the *nth-order transitional probability*.

An example of such an application of language models is in [41], it describes the use of a trigram language model with Interpolated Kneser-Ney Smoothing to create a transition matrix from the corpus. The smoothing increases low probability values like zero probabilities and decreases high probabilities, it improves the accuracy of the model. The interpolated Kneser-Ney Smoothing interpolates the trigram model with bigram and unigram models and a fixed discount is subtracted from each nonzero count. This technique also ensures that the unigram and bigram counts of a word are not just proportional to the number of occurrences of the word/bigram, instead it depends on the number of different contexts that the word/bigram follows.

1) *Trigram Language Model with Interpolated Kneser-Ney Smoothing*.: A trigram language model is used to create a transition matrix from the sample essays. This matrix keeps track of the transition probabilities from any two previous words to the next in the actual passage. For example: If *Eleanor Roosevelt's role* is the combination then

$$P(\text{role} | \text{Eleanor Roosevelt's}) = P(\text{Eleanor Roosevelt's role}) / P(\text{Eleanor Roosevelt's})$$

The various trigram, bigram and unigram counts from the sample essays are insufficient to give an accurate measure of the transition probabilities due to data sparseness. Smoothing increases low probability values, like zero probabilities, and decreases high probabilities, thereby increasing the accuracy of the model. There are various smoothing techniques [11] among which interpolated Kneser-Ney smoothing has been found to perform well in speech recognition using a perplexity metric. The interpolated Kneser-Ney Smoothing interpolates the trigram model with bigram and unigram models and a fixed discount is subtracted from each nonzero count. This technique also ensures that the unigram and bigram counts of a word are not just proportional to the number of

occurrences of the word/bigram, instead it depends on the number of different contexts that the word/bigram follows. In this model the probability  $p(w_i|w_{i-1}, \dots, w_{i-n+1})$ , which is the frequency with which word  $w_i$  occurs given that the previous  $n$  words were  $w_{i-1}, \dots, w_{i-n+1}$ , is given by

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{[c(w_{i-n+1}^i) - D], 0\}}{\sum_{w_i} c(w_{i-n+1}^i)} + \frac{D}{\sum_{w_i} c(w_{i-n+1}^i)} N_{1+}(w_{i-n+1}^{i-1} \bullet) p_{KN}(w_i|w_{i-n+2}^{i-1}) \quad (5)$$

where  $n$  for a trigram model is 3,  $c$  is a count of the number of times the n-gram  $w_{i-n+1}^i$  occurs,  $D$  is the absolute discount value set at  $n_1/n_1 + 2n_2$  where  $n_1$  and  $n_2$  are the total number of n-grams with exactly one and two counts respectively, and

$$N_{1+}(w_{i-n+1}^{i-1} \bullet) = |\{w_i : c(w_{i-n+1}^{i-1} w_i) > 0\}| \quad (6)$$

where the notation  $N_{1+}$  is meant to evoke the number of words that have one or more counts, and  $\bullet$  to evoke a free variable that is summed over.

2) *Viterbi Decoding*.: A second order Hidden Markov Model is used to infer the words of the passage. The segmented handwritten words represent the observed states and the actual words from the lexicon represent the hidden states. The top  $m$  results from the output of the word recognition for each segmented handwritten word are the possible hidden states at each time step. The state probabilities at each time step are obtained from the distance values returned by the automatic word recognition. The transition probabilities for a word given the words at the previous two time steps are obtained using the smoothing technique described previously.

Second order Viterbi decoding is used to infer the words of the passage. It evaluates the probabilities of the partial observation sequence ending at time  $t$  and the transition between every pair of consecutive states  $t - 1$  and  $t$ . It also stores the previous state at time  $t - 2$  with

the best probability in a vector and outputs the most likely sequence of words. The partial probabilities at each state for a particular word pair is given by

$$\alpha_{jk}(t) = P(s_{t-1} = j, s_t = k, O_{1..t}) = \max_i (\alpha_{i,j}(t-1) T_{ijk}) A_k(y_t), \quad (7)$$

which is the joint probability of having  $j$  as the hidden variable at step  $t - 1$ ,  $k$  at step  $t$  and observing  $O_{1..t}$  from time steps  $1..t$ . Here  $s$  refers to the hidden state i.e., the recognized character,  $T$  to the transition probability,  $A$  to the emission probability and  $t$  is the current time step. This can be expressed in terms of  $\alpha$  at time  $t - 1$ .  $T_{ijk}$  is the probability of transitioning from characters  $ij$  to  $k$  (similar to a trigram model) and  $A_k(y_t)$  is the emission probability of hidden state  $k$  emitting the observed  $y_t$  at time step  $t$ . The state corresponding to the highest probability is the most likely word in the sequence.

Word segmentation errors may lead to erroneous sequences which is handled by adding the null string to the list of  $m$  possible states at every time step. This is done to handle cases where a small part of a word has been segmented as a different word, the most likely sequence will probably identify this segment as a null string and the correct word should be identifiable from the remaining part of the segmented word. The transition probabilities are of the form

$$P(s_t | s_{t-1} = \phi, s_{t-2}) = P(s_t | s_{t-2})$$

The resulting output sequence of words are used for the scoring process.

*3) Example of Trigram processing.: An example of an essay before and after post-processing is given here. The sample essay after lexicon-based word recognition, word-spotting and their combination, is:*

lady washingtons role was hostess for the nation

first to different because lady washingtons was speeches  
for martha

and taylor roosevelt was only meetings for did people  
first vote polio on her because to see the president

The final result after contextual processing using the trigram model with  $m = 15$  is:

lady washingtons role was hostess for the nation  
but is different because george washingtons was different  
for the nation  
and eleanor roosevelt was only everything for the people  
first ladies late on her travel to see the president.

While the semantics of the text is garbled in both cases, the number of word errors is fewer after trigram processing as shown in the underlined parts. While there are still several errors in the recognized text, their effect on scoring is the true measure of performance.

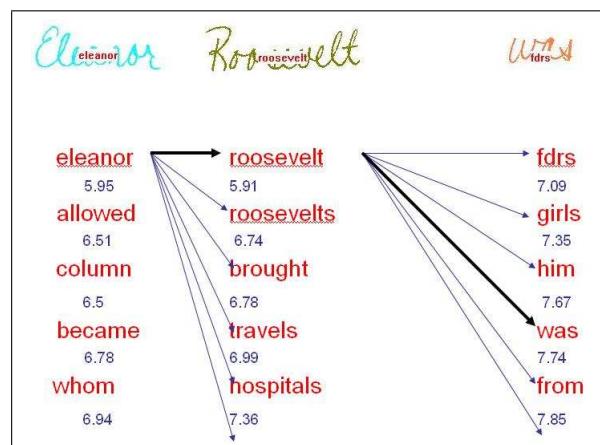


Fig. 7. Recognition Post-processing: Finding most likely word sequence

### C. Viterbi Decoding

The *Viterbi algorithm* [15] is a dynamic method of finding optimal solutions to the above quantity. Assuming that the actual word is almost always recognized in the top  $m$  choices, a Viterbi trellis with  $T$  states is built, where  $T$  corresponds to the number of segmented handwritten words in a passage. Using the n-gram approach, if there are  $T$  number of words in the answer document, then a word at position  $t$  depends on the words at position  $t - 1 \ t - 2 \dots t - n + 1$ .

In [41] a second order Hidden Markov Model is used to infer the words of the passage. The segmented handwritten words represent the observed states and the actual words from the lexicon represent the hidden states. The top  $m$  results from the output of the word recognition for each segmented handwritten word are the possible hidden states at each time step. The state probabilities at each time step are obtained from the distance values returned by the automatic word recognition. The transition probabilities for a word given the words at the previous two time steps are obtained using the smoothing technique described previously. A second order Viterbi decoding is used to infer the words of the passage. In order to account for errors in segmentation where a single word may have been segmented into two or more, the states include a null state and the transitions to state after the null state are considered from the two states preceding the null state.

### D. Rejection

The rejection strategy describe in [4] is based on a confidence measure derived from alternate candidate word sequences. The alternate candidates are derived by integrating a language model in a HMM based recognition system. Alternate candidates have commonly been generated by extracting the n-best list but candidates based on language model variations provide better rejection performance. In [4] the rejection strategy is a post processing step on the output of a

HMM based recognizer. The word sequence  $W$  with the maximum likelihood given a observed word sequence is computed using equation 8

$$\arg \max_W \log P(X|W) + \alpha \log P(W) + \beta m \quad (8)$$

where  $m$  is the number of words in  $W$ .  $P(W)$  is obtained from the language model and  $P(P(X|W))$  is obtained from the recognizer, in this case the HMM. The parameters  $\alpha$  and  $\beta$  are varied to obtain the alternate candidates. The alternative candidates are aligned with the top ranked output and each word is accepted or rejected depending on the confidence measure computed based on the alignment.

#### *E. Syntactic Tags*

A language model based on words can become intractable as the number of words in the vocabulary grows. The problem with such approaches is that as the number of words grow in the vocabulary, estimating the parameters reliably becomes difficult. More specifically, the number of low or zero-valued entries in the transition matrix starts to rise exponentially. [1] reports that of the  $6.799 * 10^{10}$  2-grams that could possibly occur in a 365,893,263 word corpus (consisting of 260,740 unique words), only 14,494,217 actually occurred, and of these, 8,045,024 occurred only once. An alternative method based on language models for word classes has been proposed in [39]. Every word is mapped onto a syntactic class like a part of Speech tag or a Named Entity Tag and the  $n$ -gram model is learned for the transition between the word classes.

In  $n$ -gram class models, words are mapped into syntactic [24], [21] (or semantic classes, discussed in the next section). In this situation,  $p(w_t | w_{t-1})$  becomes:

$$p(w_t | w_{t-1}) = p(w_t | C(w_t)) p(C(w_t) | C(w_{t-1}))$$

where  $p(C(w_t) \mid C(w_{t-1}))$  is the probability to get to the class  $C(w_t)$  following the class  $C(w_{t-1})$  and  $p(w_t \mid C(w_t))$  is the probability to get the word  $w_t$  among the words of the class  $C(w_t)$ .

The following example illustrates the use of  $n$ -gram class models by using POS tags as the method of classifying words. We use the notation  $A : B$  to indicate the case where word  $A$  has been assigned the tag  $B$ . For each sentence analyzed, a word:tag lattice is formed which representing all possible sentences for the set of word choices output by string matching (see figure 8) The problem is to find the best path(s) through this lattice. Computation of the best path requires the following information: (i) tag transition statistics, and (ii) word probabilities.

### WORD LATTICE

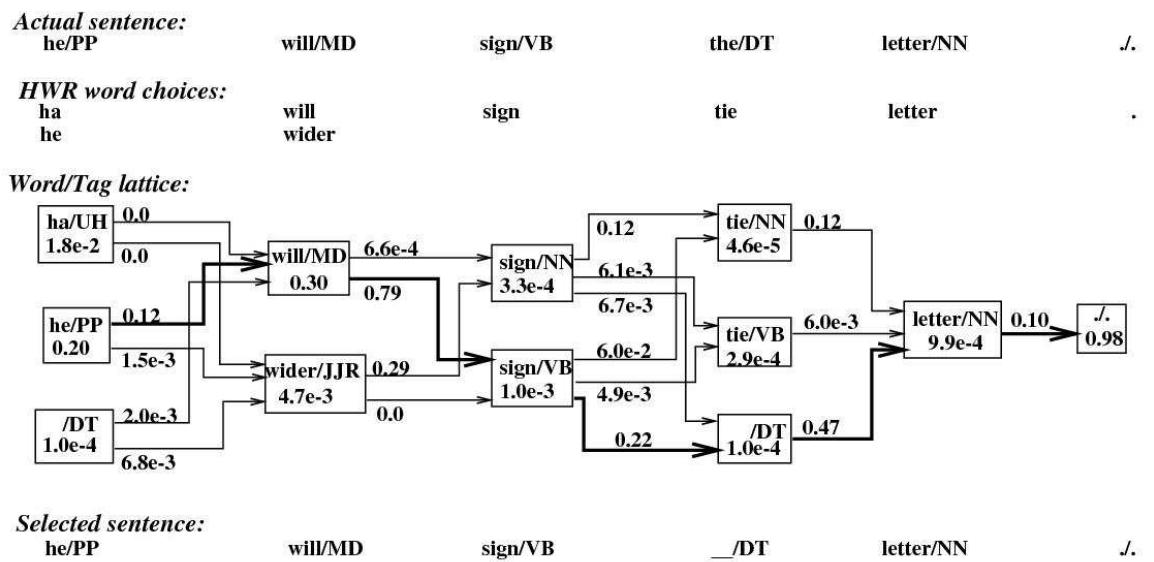


Fig. 8. Sample Word:Tag Lattice For Analysis of WR choices

Transition probabilities describe the likelihood of a tag following some preceding (sequence of) tag(s). These statistics are calculated during training as:

$$P(tag_B | tag_A) = \frac{\#(tag_A \rightarrow tag_B)}{\#(tag_A)}$$

Beginning- and end- of-sentence markers are incorporated as tags themselves to obtain necessary sentence-level information.

Word probabilities are defined (and calculated during training) as:

$$P(Word \mid Tag) = \frac{\#(Word : Tag)}{\#(AnyWord : Tag)}$$

The Viterbi algorithm is used to find the best Word:Tag sequence through the lattice, i.e., the maximal value of the following quantity:

$$\prod_{i=1}^n P(Word_i \mid Tag_i)P(Tag_i \mid Tag_{i-1})$$

over all possible tag sequences  $T = Tag_0, Tag_1 \dots, Tag_{n+1}$  where  $Tag_0$  and  $Tag_{n+1}$  are the beginning-of-sentence and end-of-sentence tags respectively. The Viterbi algorithm allows the best path to be selected without explicitly enumerating all possible tag sequences. A modification to this algorithm produces the best  $n$  sequences.

The lattice of Figure 8 demonstrates this procedure being used to derive the correct tag sequence even when the correct word ('the') was not output by the WR. The chosen path is illustrated in boldface. The values on the edges represent tag transition probabilities and the node values represent word probabilities. Although this method has been found to be effective in reducing the size of the parameters to be learnt and the size of the word neighborhood, the results were found to be not as effective as the method using the word text. This is because the method is not effective in distinguishing between word candidates belonging to the same word

class.

## V. USING LANGUAGE MODELS AT THE CHARACTER LEVEL FOR RECOGNITION

Contextual information at the character level can be expressed in terms of statistical and syntactic rules [47]. The statistical rules can be expressed as n-gram probability values. The syntactic rules include rules specific to a language, some examples are a)There cannot be 2 or more consecutive vowels b) There cannot be 3 or more consecutive consonants c) No word can end with a vowel , etc. These syntactic rules can be used to reject character sequences identified by the handwriting recognizer.

### *A. Letter ngrams in Hidden Markov Models*

Words recognizers based on Hidden Markov Models at the word level has been extensively used. At the character level each state represents a character and the transition probabilities captures the transitions probabilities between characters. In 1st order markov models this transition probability can be obtained from character level bigram probabilities. Similarly higher order HMMs use the respective n-gram probabilities. These transition probabilities can be learnt from a general corpus or dictionary or from the lexicon in a lexicon driven approach. The use of language models allows the recognition system to work without using a closed dictionaries. The language model is captured by the transition probabilities between the characters.

### *B. FSA for word recognition*

The objective here is to recognize each character image (using a classifier) and to include the language model as given by equation 2. [14] describes the use of Stochastic FSA for predictive text generation. [7] describes the difference between different SFAS that are based on Markov

model, HMM, HMM/ANN hybrid and Input/Output HMM. In [19] both transition-emitting HMMs (TE-HMMs) and state-emitting HMMs (SE-HMMs) are tailored for the purpose of modeling discrete symbols and their continuous attributes together.

The words can be represented as a regular expression using alphabets/characters. This is especially useful in languages where combinations of alphabets form characters and regular expression rules can be formed to define these combinations. A stochastic finite state automaton can be used to represent the script writing grammar.  $N$ -gram models at the character level are used to obtain a score for every possible grouping of characters to form a word image. [26] This list of possible characters can be obtained during the recognition based character segmentation process. A set of linguistic rules is applied to the SFSA model to prune out all sequence of characters which violate the rules. The SFSA models the transition between the different states and it assigns a score to every accepted word. The Viterbi algorithm is used to obtain the best possible word, the character level  $n$ -gram values are included to find the best Viterbi path. At every start state(beginning of a character) the probability of transitioning to a character given its history is included. Finite state transducers are more powerful than HMMs and can encode n-gram models efficiently. They can also encode simple grammars like spelled-out numbers One thousand two hundred and eighteen or addresses.

### *C. Error correction using Character ngrams*

[46] describes an approach to correct the OCR output using contextual information. Given a sentence to be corrected, it is broken into letter n-grams and word candidates are retrieved from the lexicon. These candidates are ranked by the conditional probability matches with the string using the character confusion probabilities. The conditional probability  $p(s|w)$  is the probability that the word  $w$  is identified as  $s$  by the recognition system. A dynamic programming algorithm

is used to determine the operations (insert character, delete character and substitute character) that maximizes the conditional probability. The character confusion probabilities account for a character  $x$  being recognized as  $y$  or for  $xy$  being recognized as  $z$ , these probabilities can be learnt independently. The lexicon entries are indexed by their character ngrams. The word candidates are the lexicon entries which share a large number of character ngrams in common with the recognized word. After obtaining the top  $N$  candidates, the error correction techniques at the word level described in section IV can be used. This correction can be applied to all words or only to those words which do not match an entry in the lexicon depending on the performance of the recognizer.

## VI. LEXICAL AND SYNTACTIC ANALYSIS

### A. Lexical Analysis Using Collocational Information

Collocational information [37] can be used in order to modify word neighborhoods generated by the WR. These modified neighborhoods are then input to a statistical syntax analysis module which makes final word choices. Collocations are word patterns that occur frequently in language; intuitively, if word A is present, there is a high probability that word B is also present. Collocations are categorized based on (i) the strength of their association (mutual information score,  $mis$ ) and (ii) the mean and variance of the separation between them.

Specifically, collocational analysis can result in the following actions (ranked from conservative to aggressive):

- 1) re-rank the word choices thereby promoting more likely words,
- 2) eliminate word choices thereby reducing word neighborhoods, or
- 3) propose new words (not in the top  $n$  choices of the WR).

The first two actions are possible only if, for each word in the multi-word collocation, the correct

word is among the top  $n$  choices of the WR. The last action does not have this restriction and constitutes a form of error detection and correction.

Based on the (i) the strength of the collocation that a word choice participates in,  $mis(xy)$ , and (ii) the confidence given to this word by the WR  $wr\_conf(x)$ , a decision is made whether to simply promote a word choice (i.e., increase its rank) or to promote it to the top choice and eliminate all other word choices for each of the word neighborhoods participating in the collocation.

Collocational analysis can be extended by using one-sided information score. For example, the word ‘offended’ is frequently followed by the word ‘by’, but the word ‘by’ may be preceded by virtually anything. Such analysis extends the utility of collocational analysis but comes with a risk of promoting incorrect word choices.

### *B. Syntactic Analysis*

Syntactic models are the most common language model used in post processing the output of noisy channels (speech, OCR). Methods to apply syntactic knowledge to this problem fall into the following categories:

- $N$ -gram word models
- $N$ -gram class (e.g., part-of-speech) models
- Context-free grammars
- Stochastic Context-free grammars

Context-free grammars model language by capturing the syntax of all possible input sentences and reject those sentence possibilities which are not accepted by the grammar. The problems with such a method are (i) the inability of the grammar to cover all possibilities (especially since informal language is frequently ungrammatical) and (ii) the computational expense involved in

parsing (increased due to several word choices). Stochastic CFG's [28] alleviate the first problem somewhat since techniques have been developed for automatically constructing CFG's from large training corpora. Ngram word models were discussed in section IV-B and ngram class models were discussed in section IV-E.

### *C. Hybrid Method*

While the previous method is effective in reducing word neighborhood sizes, it does not seem to be effective in determining the correct/best sentence (the ultimate objective) or in providing feedback. A hybrid method attempts to overcome this, it combines both analytical and statistical knowledge about syntax. There are several problems which the previous method has. First, is the inability to capture long-distance phenomena such as subject-verb agreement. For example if the sentence were "The leader of our group *give* a talk", and the correct word *gave* was an alternate choice at the sixth word position, the POS bigram/trigram method would fail to choose it. Although the speech recognition community has tried to overcome such problems by using long-distance  $n$ -grams, it is computationally expensive and not always successful. Second, we desire another source of information for spotting errors made by the recognizer. Hypertagging involves partitioning sentences into meaningful syntactic units based on the POS associated with each word. The hypertagging bigram method can be combined with the POS bigram method by modeling the problem as a variable duration markov model.

## VII. INFORMATION EXTRACTION.

An important role can be played by information extraction (IE) in handwriting recognition. IE from unstructured data or natural language text has made tremendous advances making possible applications such as natural language question answering and intelligent document retrieval. IE

allows the extraction of parts of speech, named entities and concepts from a given corpus or passage of text. Such information is essential to not only assess the quality of the answers but also to improve the performance of the recognizer itself.

IE provides the who, when, where and how much in a sentence. In addition the relationships between entities, and entity profiles including modifiers and descriptions, e.g., “hostess for the nation” can be identified. An example of the result of IE processing for a handwritten student essay is given in Figure 9. This is an output screen of the Semantex<sup>TM</sup> engine which shows that (i) “Marta Washington” is detected as a person entity, and (ii) “Eleanor Roosevelt” is not a co-reference, i.e., is not the same person. To handle spelling and recognition errors the profiles for “Martha Washington” and “Marta Washington” could be merged by noting spelling and semantic similarities.

Named entity (NE) tagging is one of the core reasons to use an IE engine. NEs are the atomic elements of language as a whole are considered to be the *who*, *where*, *when* and *how much* in a sentence. An NE is a word or phrase that denotes a proper name such as person, organization, location, product, temporal expression and numerical expression. The expectation of a given word or phrase in a location can narrow down the lexicon thereby improving recognition.

### VIII. MULTILINGUAL LANGUAGE MODEL

We briefly discuss the use of language models in text recognition in Devanagari, Arabic and Chinese. The work done in recognition using language models in these languages is considerably lesser than that done for the Latin script [42], [44]. Hardly any language model work is reported for Arabic [42].

The inclusion of language models into these scripts may appear to be more challenging than that for Latin as the character sets are much larger. However these scripts are more structured

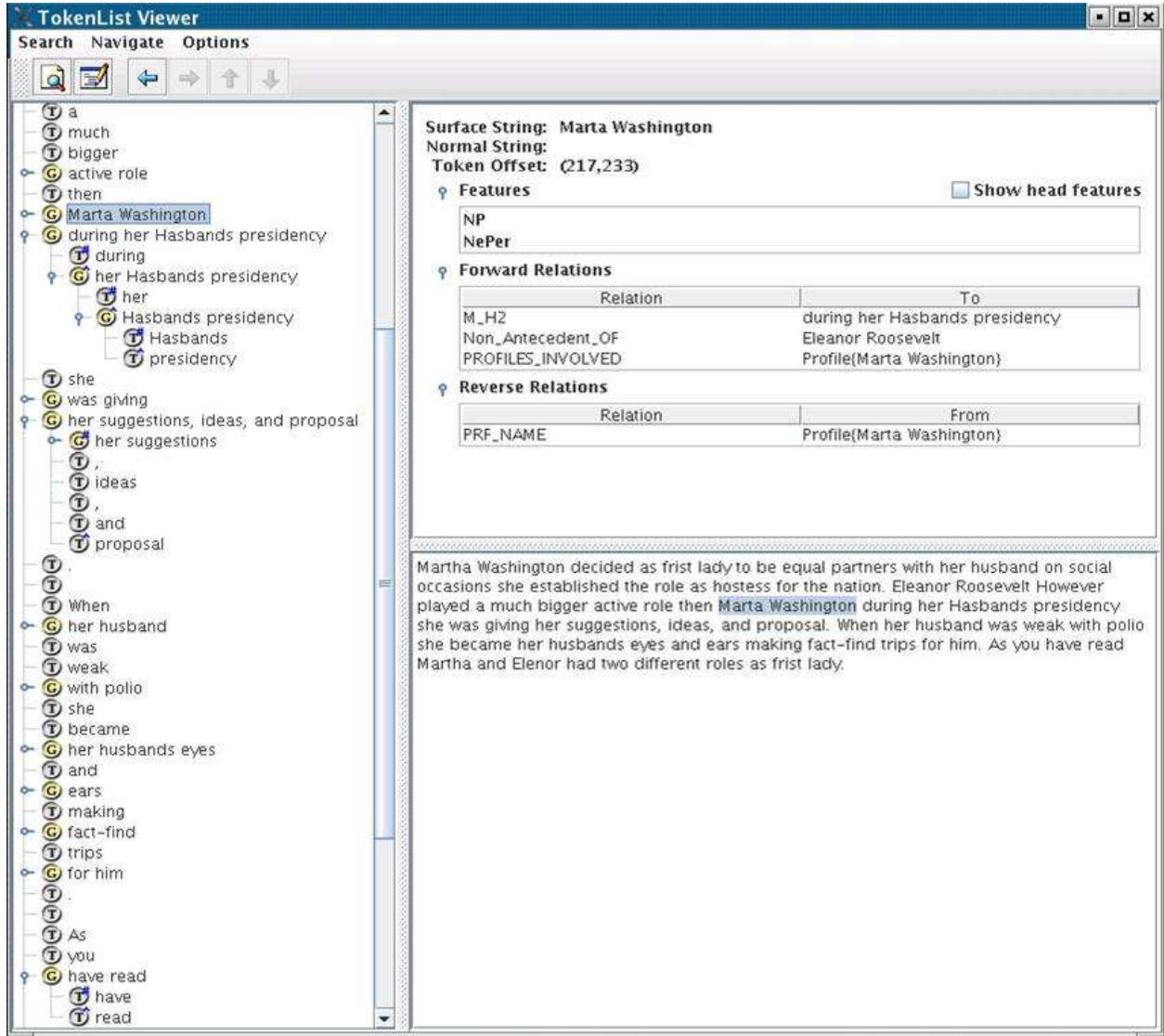


Fig. 9. Information Extraction result of processing a transcribed student essay by Semantex™. The lower right-hand pane shows the essay being processed with “Marta Washington” highlighted. The left-hand pane shows the partial token list that gives the syntactic parse and semantic features where “Marta Washington” is identified as a group (G). The right-hand top pane shows the profile of “Marta Washington” which is identified as a noun phrase with named entity tag of “person” who is different from “Eleanor Roosevelt”.

and have a lower level of ambiguity of words which could lead to a better language model. [26] proposes a techniques for word recognition for Devanagari script. A stochastic finite state automaton can be used to represent the script writing grammar.  $N$ -gram models at the character level are used to obtain a score for every possible grouping of characters to form a word image. [3] describes the correction of optically read Devanagari character sequences using a

Hindi word dictionary. Script writing rules are used to narrow the list of possible characters. [30],[31] investigated several language models in the contextual post-processing of Chinese script recognition. They gave the proposal in choosing a suitable language model according to the requirement of a practical recognition system. [29] uses a trigram Arabic language model to determine the most probable morpheme sequence for a given input. The language model is first learnt from a small manually segmented corpus and then the model parameters are learnt using an unsupervised algorithm for automatically acquiring new stems from a much larger word unsegmented corpus.

## IX. CONCLUSION

Language modeling has been found to be very helpful for all natural language applications. The application of language models to the problem of text recognition has been discussed here for two different paradigms - word level and character level. The application of language models at all stages of recognition are described including the integration of word, character and syntactic models in the recognition phase, lexicon filtering stage and the syntactic and semantic post-processing of the recognition output. The post-processing improves the recognition results but it has been shown that the utility of post-processing decreases as the word recognition improves. The potential improvement from post-processing is superceded by the better performance of the recognizer. For it to be effective in text recognition it is important that the text used in training is representative of that for which the model is intended. The use of language models trained on an irrelevant corpus can possibly make the recognition worse. The higher-level language modelling techniques are applicable to other languages as well as other modalities (such as machine-print, speech recognition).

## REFERENCES

- [1] Averbuch et al. Experiments with the tangora 20,000 word speech recognizer. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 701–704, 1987.
- [2] 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):179–190, 83.
- [3] V. Bansal and R.M.K. Sinha. Partitioning and searching dictionary for correction of optically-read devanagari character strings. In *International Journal on Document Analysis and Recognition*, volume 4, pages 269–280, 2002.
- [4] R. Bertolami, M. Zimmermann, and H. Bunke. Rejection strategies for offline handwritten text line recognition. *Pattern Recognition Letters*, 27:2005–2012, 2006.
- [5] R. Bertolami and H. Bunke. Multiple handwritten text line recognition systems derived from specific integration of a language model. In *Eighth International Conference on Document Analysis and Recognition*, pages 521–527, 2005.
- [6] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [7] H. Bourlard and S. Bengio. Hidden markov models and other finite state automata for sequence processing. In *The Handbook of Brain Theory and Neural Networks, Second Edition*. Cambridge, MA: The MIT Press, 2002.
- [8] T. M. Breuel. Language modeling for a real-world handwriting recognition task. In *Computational Linguistics for Speech and Handwriting Recognition*, 1994.
- [9] P.F. Brown, P.V. deSouza, R.L. Mercer, V.J. Della Pietra, and J.C. Lai. Class-Based  $n$ -gram Models of Natural Language. *Computational Linguistics*, 18(4):467–479, 1992.
- [10] M. Chen, A. Kundu, and S.N. Srihari. Handwritten Word Recognition using Variable Duration Hidden Markov Models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP-93)*, pages (5):105–108, 1993.
- [11] S.F. Chen and J. Goodman. Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998.
- [12] K. Church. A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text. In *Proceedings of the Second Conference on Applied Natural Language Processing (ACL)*, pages 136–143, 1988.
- [13] J. Favata. *Recognition of Cursive, Discrete and Mixed Handwritten Words using Character, Lexical and Spatial Constraints*. PhD thesis, State University of New York at Buffalo, Dept. of Computer Science, 1993.
- [14] M.L. Forcada. Corpus-based stochastic finite-state predictive text entry for reduced keyboards: application to catalan. In *Procesamiento del Lenguaje Natural*, volume 27, pages 65–70, 2001.
- [15] G. E. Forney Jr. The Viterbi Algorithm. *Proceedings of IEEE*, 61:268–278, 1973.

- [16] W.A. Gale and K.W. Church. Poor Estimates of Context are Worse than None. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 283–287, 1990.
- [17] R. Garside, G. Leech, and G. Sampson. *The Computational Analysis of English*. Longman, 1987.
- [18] I.J. Good. The Population Frequencies of Species and the Estimation of Population Parameters. *Biometrika*, 40:237–264, 1953.
- [19] X. Hanhong and V. Govindaraju. Hidden markov models combining discrete symbols and continuous attributes in handwriting recognition. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, pages 458–462, 2006.
- [20] J. Hull. *A Computational Theory of Visual Word Recognition*. PhD thesis, University at Buffalo Department of Computer Science (CEDAR), 1988.
- [21] J.J. Hull. Incorporation of a Markov Model of Syntax in a Text Recognition Algorithm. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 174–183, 1992.
- [22] F. Jelinek. *Statistical Aspects of Speech Recognition*. MIT Press, 1994.
- [23] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. A dynamic language model for speech recognition. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 293–295, 1991.
- [24] F.G. Keenan, L.J. Evett, and R.J. Whitrow. A large vocabulary stochastic syntax analyser for handwriting recognition. In *Proceedings of the First International Conference on Document Analysis (ICDAR-91)*, pages 794–802, 1991.
- [25] G. Kim. *Handwritten Word and Phrase Recognition*. PhD thesis, Department of Computer Science (CEDAR), State University of New York at Buffalo, 1996.
- [26] S. Kompalli. *A Stochastic Framework for Font-Independent Devanagari OCR*. PhD thesis, University at Buffalo Department of Computer Science and Engineering (CEDAR), 2007.
- [27] K. Kukich. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4), 1992.
- [28] K. Lari and S.J. Young. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4(1):35–56, 1990.
- [29] Y. Lee, K. Papineni, and S. Roukos. Language model based arabic word segmentation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 399–406, 2003.
- [30] Y. Li and C.L. Tan. An empirical study of statistical language models for contextual post-processing of chinese script recognition. In *Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*, volume 1, page 257262, 2004.
- [31] Y. Li and C.L. Tan. Influence of Language Models and Candidate Set Size on Contextual Post-processing for Chinese Script Recognition. In *Proceedings of the 17th International Conference on Pattern Recognition*, volume 2, page 537540,

- 2004.
- [32] D.M. Magerman and M.P. Marcus. Parsing the Voyager Domain using Pearl. In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 231–236, 1991.
  - [33] U. Marti and H. Bunke. On the Influence of Vocabulary Size and Language Models in Unconstrained Handwritten Text Recognition. In *Proceedings of the Sixth International Conference on Document Analysis and Recognition*, pages 260–265, 2001.
  - [34] U.V. Marti and H. Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1):65–90, 2001.
  - [35] G. Seni, K. Sundar, and R.K. Srihari. On the use of edit distance for handwritten text recognition. In *Proceedings of the SPIE conference on Document Understanding (to appear)*, San Jose, CA, 1995.
  - [36] G. Seni, K. Sundar, and R.K. Srihari. Generalizing Edit Distance for Handwritten Text Recognition. *Pattern Recognition*, 29(3):405–413, 1996.
  - [37] F. Smadja. Macrocoding the Lexicon with Co-Occurrence Knowledge. In Uri Zernik, editor, *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon*, pages 165–189. Lawrence Erlbaum Associates, Hillsdale, NJ, 1991.
  - [38] R. K. Srihari, C. Ng, C. Baltus, and J. Kud. Use of language models in on-line recognition of handwritten sentences. In *Proc. Third International Workshop on Frontiers in Handwriting Recognition, Buffalo, NY*, pages 284–294, 1993.
  - [39] R.K. Srihari. Combining ststistical and syntacting techniques in recognizing handwritten sentences. In *Probabilistic Approaches to Natural Language, AAAI Symposium, Cambridge, MA*, pages 121–127, 1992.
  - [40] R.K. Srihari. Use of lexical and syntactic techniques in recognizing handwritten text. In *Proceedings of ARPA workshop on Human Language Technology, Princeton, NJ*, pages 403–407, 1994.
  - [41] S. Srihari, R. Srihari, H. Srinivasan, and P. Babu. On the automatic scoring of handwritten essays. In *Proceedings of International Joint Conference on Artificial Intelligence, Hyderabad, India*, pages 2880–2884, 2007.
  - [42] S. N. Srihari and G. R. Ball. Arabic Handwriting Recognition: An Assessment. Technical report, CEDAR SUNY at Buffalo, 2007.
  - [43] S. N. Srihari, A. Ganesh, C. Tomai, Y.C. Shin, and C. Huang. Information retrieval system for handwritten documents. In *Document Analysis Systems VI: Proc. 6th International Workshop, Florence, Italy*, pages 298–309. Springer LNCS, 2004.
  - [44] S. N. Srihari, X. Yang, and G. R. Ball. Offline Chinese Handwriting Recognition: A Survey. *Frontiers of Computer Science in China*, 1(2), 2007.
  - [45] S.N. Srihari, H. Srinivasan, C. Huang, and S. Shetty. Spotting words in Latin, Devanagari and Arabic scripts. *Vivek: A Quarterly in Artificial Intelligence*, 16(3):2–9, 2006.

- [46] X. Tong and D. Evans. A statistical approach to automatic ocr error correction in context. In *Proceedings of the fourth workshop on very large corpora*, pages 88–100, 1996.
- [47] G. T. Toussaint. The Use of Context in Pattern Recognition. *Pattern Recognition*, 10:189–204, 1978.
- [48] A. Vinciarelli, S. Bengio, and H. Bunke. Offline recognition of unconstrained handwritten texts using HMM and statistical language models. Technical report, IDIAP-RR 03-22, Dalle Molle Institute for Perceptual Artificial Intelligence, 2003.
- [49] R.A. Wagner and M.J. Fischer. The String-to-String Correction Problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, 1974.
- [50] M. Zimmermann and H. Bunke. N-gram language models for offline handwritten text recognition. In *Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR'04)*, pages 203–208, 2004.