# Word Prediction via a Clustered Optimal Binary Search Tree

Eyas El-Qawasmeh
Computer Science Department, Jordan University of Science and Technology, Jordan

**Abstract:** *Word prediction methodologies depend heavily on the statistical approach that uses the unigram, bigram, and the trigram of words. However, the construction of the N-gram model requires a very large size of memory, which is beyond the capability of many existing computers. Beside this, the approximation reduces the accuracy of word prediction. In this paper, we suggest to use a cluster of computers to build an Optimal Binary Search Tree (OBST) that will be used for the statistical approach in word prediction. The OBST will contain extra links so that the bigram and the trigram of the language will be presented. In addition, we suggest the incorporation of other enhancements to achieve optimal performance of word prediction. Our experimental results showed that the suggested approach improves the keystroke saving.*

**Keywords:** *Bigram, cluster computing, N-gram, unigram, trigram, word frequency, word prediction.*

*Received April 21, 2003; accepted July 29, 2003*

## 1. Introduction

In general, word predication is the problem of guessing the next word in a sentence as the sentence is being entered, and updates this prediction as the word is typed. Currently "word prediction" implies both "word completion and word prediction" [13]. Word completion is defined as offering the user a list of words after a letter has been typed, while word prediction is defined as offering the user a list of probable words after a word has been typed or selected, based on previous words rather than on the basis of the letter. Word completion problem is easier to solve since the knowledge of some letter(s) provides the predictor a chance to eliminate many of irrelevant words [4, 13].

"The task of prediction the most likely word based on properties of its surrounding context is the archetypical prediction problem in Natural Language Processing (NLP). In many NPL tasks, it is necessary to determine the most likely word, part-of-speech (POS) tag or any other token, given its history or context. Examples include part-of-speech tagging, word-sense disambiguation, speech recognition, accent restoration, context-sensitive spelling correction, and identifying discourse markers" [4]. Currently, word prediction is used in many real life applications such as augmentative communication devices [12].

The development of more sophisticated prediction techniques can provide high degree of keystroke saving (percentage of keystrokes eliminated by integrating the prediction method) which may translate to faster communication rates [14]. In addition, it can improve the quality (as well as the quantity) of message production for persons with language impairment, communication impairments, and those with learning disabilities [13].

Currently, there are many approaches, which have been developed for use in word prediction. These approaches can be classified into three groups: (1) use of statistical calculations, which can be either dynamically adaptable/not adaptable to their user; (2) use of syntactic information in order to improve the accuracy of the prediction; (3) use of semantics of the sentence to help the prediction process [15].

In this paper the difficulty in expressing the statistical model without losing any accuracy will be explained. As an alternative we suggest to build the N-gram frequency of the statistical model using a cluster of computers. Thus, an OBST with some extra links will present the unigram, bigram, and the trigram collectively. Once the N-gram has been built, we will use the statistical model so that more accuracy will be gained. This will be followed by integrating other techniques to improve the suggested structure.

The organization of this paper will be as follows. Section 2 presents the basic concepts of the prediction methodology. Section 3 is a discussion of the language model. Section 4 presents the suggested structure for the statistical model. Section 5 will be the integration of other enhancements to the suggested structure. Section 6 explains the results of the experimentations that were carried out; and finally section 7 draws conclusions and future work.

## 2. Basic Concepts

The majority of current word prediction systems employ statistical analysis for their word prediction. "The choice of words for placement in the prediction

list is based upon the probability that they will appear in the text. The probabilities can be fixed, based on the language in general, or they can be based on the user's own style and altered as the system is used. In the simplest systems, the probabilities relate only to isolated words and their likehood of use. In more complex approaches, the statistics are based upon the probability of a word appearing given what has gone before" [15].

The easiest way of word prediction is to use a fixed lexicon. In this case, each word in the lexicon will have a frequency associated with it that relates the word to how often it is used in the language in general. These frequencies are based on large textual or written documents like news papers, and even sometimes from spoken language.

Following is a formal definition of the prediction, which generates a list of words for the user.

*Definition:* Let *A* be a finite language that consists of words $X_1$, $X_2$, $X_3$,.. , $X_n$ . Given any sentence, which consists of a set of words $w_1$, $w_2$, $w_3$,…. A prediction applied to this sentence will generate a sequence of words C = $X_1$`, $X_2$`, $X_3$`, in which $X_i$` is the predicted value of $X_i$ for every i.

Two different methods are used to produce the predictions. The first method sort the whole lexicon according to the frequency order and offer the user some few words from the prediction list with the highest frequencies. If the prediction, which is required, does not appear on the list straight away then the user types in the first letter. The list is then reduced to only those words which have the initial letter just entered, and again, the top few are offered as predictions. This process continues until either the word has been spelt out in its entirety, or it has appeared on the prediction list, at which point the user can add it to the sentence in whichever way is made available to him [15].

The second method of using a fixed lexicon requires more detailed corpus data. "Words stored in the dictionary are tagged with the frequencies with which they appear after other given words. Consequently, when a word is entered, the most frequently used word which follows it can be extracted from the corpus to produce a predication list. The advantage of this method over the previously discussed method is that the prediction list will be much more on the current status of the sentence and therefore more likely to contain correct predictions. This makes it a far more common choice for designers of prediction systems. The process of selecting a word is the same as for the previous method, although the list, which is initially drawn up, is changed each time a new word is entered rather than being a fixed list as was seen before" [15].

## 3. Language Model

Language models for word prediction concentrate exclusively on the probabilities of the words in the history [2]. In this paper, we suggest to construct an OBST so that its goal is to find the best sequence of words that can be used in word prediction. To build the OBST, a cluster of computers will be used. The cluster of computers allows handling the big size of different combinations. We will show that our structure reduces the word error rate and improves the Keystroke Saving Rate (KSR) for certain domains of a corpus when it is used with other prediction approaches.

Given a sequence of words W=$w_1$ $w_2$….$w_N$, the probability language model estimate the probability of a sequence of words W using Bayes rule as the product of conditional probabilities as follows:

$$Pr(W_{1,N}) = \prod_{i=1}^{N} Pr(W_i \mid W_{1,i-1}) \quad (1)$$

where N represents number of words and it is a random variable itself. Equation (1) computes the probability of observing a word $w_i$ at position i; modeled as being restricted to its immediate (i-1) predecessor words $W_{1,i-1}$. The result is a Markov chain, called an N-gram model [18]. N-gram models have been among the most successful approaches used for language modeling. These refer to finite state analysis of series of 1, 2, or 3 words sequences, such as "doors", "building doors", or "like building doors", and are called unigram, bigram, and trigram.

To compute the probability distribution the traditional way is to define equivalence classes amongst the contexts $W_{1,i-1}$ which can be done by limiting the contexts to an N-gram language model [9, 16]. One can also mix in smaller size language models when there is not enough data to support the large context by using either interpolated estimation [10] or a backoff application [11].

Following is the equation that is usually used for a class-based trigram model, where the function $g$ maps each word to its unambiguous class

$$Pr(W_i \mid W_{1,i-1}) = Pr(W_i \mid g(W_i) Pr(g(W_i) \mid g(W_{i-1}) g(W_{i-2})) \quad (2)$$

Using classes has a potential problem such that the class for adjacent words $g(w_{i-1} \mid w_i)$ and $g(w_i)$ lose some information about each other. This loss of information occurs in many algorithms such as Brown algorithm [2].

Below we give the derivation based on using the trigram:

$$Pr(W_{1,N}) = \sum_{P_{1,N}} Pr(W_{1,N} P_{1,N}) \quad (3)$$

$$= \sum_{P_{1,N}} \prod_{i=1}^{N} Pr(W_{i|} \mid W_{1,i-1} P_{1,i}) Pr(P_i \mid W_{1,i-1} P_{1,i-1}) \quad (4)$$

$$= \sum_{P_{1,N}} \prod_{i=1}^{N} Pr(W_i \mid P_i) Pr(P_i \mid P_{1,i-1}) \quad (5)$$

$$= \sum_{P_{1,N}} \prod_{i=1}^{N} Pr(W_i \mid P_i) Pr(P_i \mid P_{i-2,i-1}) \quad (6)$$

Note that equation (4) involves some simplifying assumptions, namely that $\Pr(W_i \mid W_{1,i-1}P_{1,i})$ can be approximated by $\Pr(W_i \mid P_i)$ and that $\Pr(P_i \mid W_{1,i-1}P_{1,i-1})$ can be approximated by $\Pr(P_i \mid P_{1,i-1})$.Although those assumptions simplify the task of estimating the probability distributions; they reduce the accuracy of the current model. The above approach when incorporated in some applications such as speech recognition does not improve the performance. Srinivas [17] reported an increase in the *perplexity*, which is a measure, over a word-based model on the Wall street Journal by 24.5%. Another technique for the construction of n-gram language model is the one that is developed by Chen and others [3]. However, this scheme is an instance of Jelinek-Mercer smoothing approach [3], therefore, we will not go into further details of it.

The main limitation of the previous language model is that the number of states and transitions is unmanageable. As a result, the use of equivalence classes and the approximation reduces from the accuracy of the word prediction. In addition, the use of a single computer does not offer the capability to build the trigram. Therefore, we suggest building the trigram using a cluster of computers. For example, consider a vocabulary of size V then there will be $V^3$ possible trigrams, which for 20,000 words translates to 8 trillion trigrams. This number cannot be handled by a single computer. However, a cluster of computers can handle this size. In addition, our structure will not reserve any space for many trigrams that will not be seen in the training corpus (have zero probability). In the case the user does not like the sparness in this model then he can avoid this by using tagging instead of classes in language modeling [7].

## 4. Suggested Structure

The construction of the trigram requires a large amount of memory. Therefore, a cluster of computers will be used to build this model. To achieve optimal performance we suggest using the OBST. This OBST will contain extra links so that the bigram, and trigram will be included in this structure.

In our suggested approach, a cluster of PC's connected via Ethernet connection is used to build the frequencies. The number of the hosts was equal to the number of the language letters although it is possible that the number of hosts is less than the letters. The MPI software was used to implement this configuration. In the implemented configuration, one of the computers was the master and the others were the workers, and the MPI software control the interchange of the messages depending on the availability. The master PC starts processing the corpus, and then it starts sending messages to the hosts. Each host was responsible for a single letter in the language. Thus all the words that start with this letter will go to its designated processor. The host PC will be responsible for storing the word and keeping track of the frequencies of all the words that start with that letter.

At this point we are able to get the unigram of the words distributed among several hosts. After that our program starts building an OBST for each PC. The master computer has the links to al other hosts and it has the capability to access all the OBSTs. It should be noted that our algorithm tries to make the OBST as a balanced tree as possible meanwhile taking into consideration that the load is distributed among the hosts [1]. In case the load is not distributed well, then it is possible to merge two rare beginning letters of words into one group.

Each node in this optimal binary search tree will have five attributes. They are the word itself, its frequency, the left and right child pointers, and a pointer value to a linked list. The linked list represents the bigram for all the words that start with the word that is already residing in the OBST. Note that the selection of a binary search tree implies that the most frequent word(s) will be close to the root. In addition, all the words in the left sub-tree are less than or equal to the root of the sub-tree, and all the nodes on the right sub-tree are greater than or equal to the sub-tree root [6]. This data structure has the advantage in that the search will take O(h) where h is the height of our binary search tree. As an example, for vocabulary of 60,000 words, the number of visited nodes from the root to the leaf will take at most 16 comparisons since the $\log_2$ (Number of words) is at most equal to ( $\log_2$ 60000) $\approx$ 16.

To complete the construction of our OBST, we will revisit each node and identify the word it represent and start the creation of the bigram. For example, let us assume that the root contains the word "the", Then this word will be checked all over the corpus and the appearance of two adjacent words where the first word is "the" will be tracked. In other words, every two-word combinations in the language where the first word is "the" will be counted. Doing this will allow us to identify all dual words where one of them is exactly the word which is stored in the node. Upon determining them we will be able to construct a sorted linked list based on the frequency. This sorted linked list varies in size from one node to another. In the worst case, a word like "the" will have a long linked list of size approximately equal to the number of nouns in English, while other nodes might have a short nodes

that consist of few nodes. The same thing appears in all languages. For example, a node that represents a value of "I" might have a small linked list of at most one or two items. Following is Figure 1, which gives a general sketch of the tree.

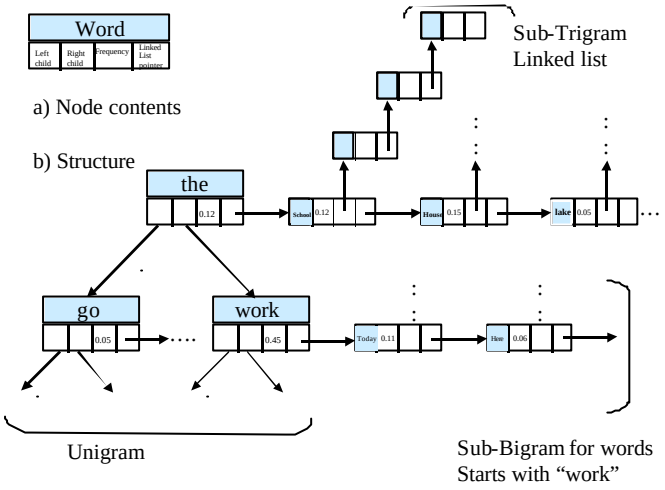



Figure 1. General structure of the optimal binary search tree with extra links.

It should be noted that although we report the frequency in the previous figure, but we can eliminate it for the purpose of saving memory. In fact, after we get the frequency for each linked list, we used to sort and re-structure the linked list for the bigram, and then for the trigram in order to eliminate the frequency field of the nodes.

Following is the exact algorithm. It consists of two stages. The static stage, which is executed only once before the user starts using the system and the dynamic stage where the user can get the predicated word.

### *a. Static (Creating) Stage*

1. *Broadcast Step:* Scan a corpus. For each sequence of words $w_i$ $w_{i+1}$ $w_{i+2}$ send a message to the processor which takes care of the words that start with the first letter of word $w_i$. The designated processor will keep track of the frequencies for the for the unigram $w_i$, bigram $w_i$ $w_{i+1}$ and trigram $w_i$ $w_{i+1}$,$w_{i+2}$.
2. *Bigram Combine Step:* Build an optimal binary search tree with five identifiers. The first identifier is the word. The second is the frequency. Other identifiers are designed for the left and right child pointers of the node, and the fifth identifier is a pointer to a linked list which will be created following. This linked list represents a sub-bigram for a single word.
3. *Trigram Combine Step:* Use the clustered bigram table to build the linked list for each word. The linked list will be sorted according to the frequency with the highest frequency at the front of the linked list.

### *b. Dynamic (Run) Stage*

1. Read a sentence with a missing word (let the missing word be denoted by $w_j$).
2. Let C ← ϕ
3. Find the *previous* word ($w_j$) and the *next* word ($w_j$).
4. *if (previous ($w_j$) = NULL and next ($w_j$) ¹ NULL ) then*
   *begin*
   *for each node $w_j$ in our OBST do*
   *Visit the corresponding linked list of the bigram.*
   *if next ($w_j$) Î the linked list of $w_j$ then*
   *C ¬ C È $w_j$ where $w_j$ is the word stored in the OBST.*
   *Apply Further_ Filtering on set C.*
   *end*
   *else*
   *begin*
   *if (previous ($w_j$)¹ NULL and next ($w_j$)= NULL) then*
   *Visit previous ($w_j$) which already exists in the OBST. From there visit the corresponding linked list attached to it and let C ¬ C È $w_i$ | where $w_i$ Î linked list of (previous ($w_j$)) and $w_i$ is unique.*
   *Apply Further_Filtering on set C.*
   *else*
   *Visit previous ($w_j$). From there visit the corresponding linked list attached to it and obtain the set Candidates C such that C = C È $w_{j-1}$ | where $w_j$ is within the sequence $w_{j-1}$ $w_j$ $w_{j+1}$*
   *Generate C = C Ç next ($w_j$) | $w_j$ within the sequence of $w_{j-1}$ $w_j$ $w_{j+1}$.*
   *Apply Further_Filtering on set C.*
   *end if*
   *end*
4. Use suggested improvement in the following section to reduce size of set C.
5. Present the list as a suggested list for the user.

We run the implementation of our suggested data structure on a cluster of 26 machines using MPI software for a specific domain from corpus. Our observations showed that it will be a useful structure in many specialized domains like using word prediction in phrases from different religious holy books.

## 5. The Integration of Other Enhancement to the Suggested Structure

The goal of this research was not to replace the process by which prediction is chosen, but to augment it by reducing the domain of search used by existing methods. Thus, the use of our structure combined with

other enhancement methods will improve the overall word prediction.

As can be seen from our algorithm, it calls a procedure named "Further-Filtering". The Further_Filtering procedure has numerous techniques to enhance N-gram word prediction using recency, syntactic analysis, syntax-based N-gram, and domain-specific N-gram models. Other enhancements can be considered such as dividing the 1st order frequency table into sub-tables, for example, one will be used for verbs, and the other will be used for nouns and so on. These enhancements can be defined inside this procedure. However, because they are out of the scope of this paper, we did not describe them. As an alternative, we used another word predictor (Soothsayer) behind our program at some points of the experimentation results.

## 6. Experimentation Results

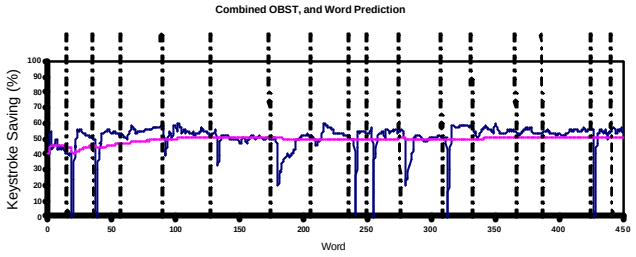
The keystroke saving was the main test used for our structure. This test investigates the savings, which a prediction system can offer the user. It is given as one minus the ratio of key strokes actually made to the number of letters in the text. The higher this figure is, the more savings are available to the user, and consequently, the better the system is.

To test the performance, a piece of medical text was selected. We run four cases for word prediction using the same text. The first case (Figure 2, part a ) uses our structure combined with another word prediction program. The second case (Figure 2, part b) uses our structure, without any other word prediction program. The third case (Figure 2, part c) one is the word prediction "soothsayers" alone. The final case (Figure 2, part d) uses no prediction at all, instead creating a prediction list which contains the entire lexicon. The keystroke saving throughout each sentence was recorded and shown in the figures. On each graph, the main plot indicates the keystroke saving for each sentence, plotted word by word. This plot is reset to zero at the beginning of each sentence. In order to clarify the effect of sentence boundaries, vertical dashed lines mark them. The final information added to these groups is a plot of the cumulative keystroke saving over the entire text. This gives a useful indication of the overall performance of our structure.

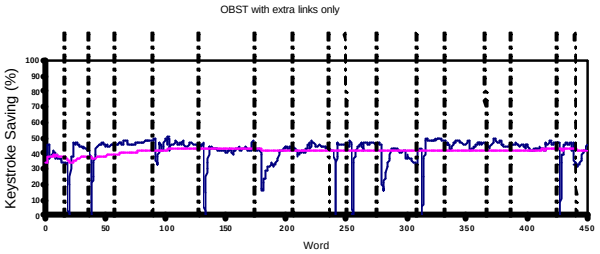
The overall keystroke saving can be seen in Figure 3. Results of this figure showed that if we use our structure combined with other word prediction techniques, then an increase in saving of keystroke is approximately 20%.

## 7. Conclusion

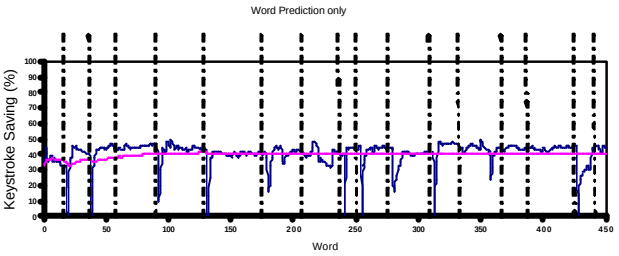
Word prediction can be used in many applications. For example, it can be used to disambiguate sequences from ambiguous keypads, correct spelling errors, and provide an OBST with some extra links to be used and built using a cluster of computers. This can be used for certain domain-specific topics such as medical topics. It provides more accuracy, and a keystroke saving of around 20% when combined with other approaches.



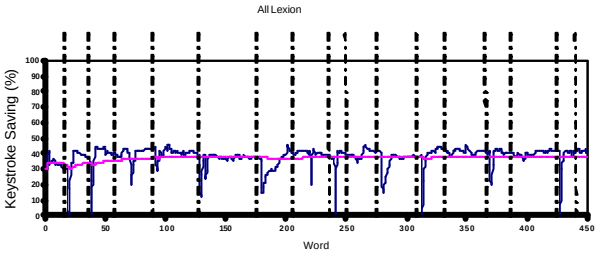

a) OBST combined with word prediction program.




b) OBST with extra links.




c) Ready word prediction program only.




d) Retrieval of whole lexicon.

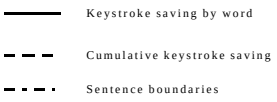



Figure 2. Keystroke saving for a medical article using various prediction algorithms.

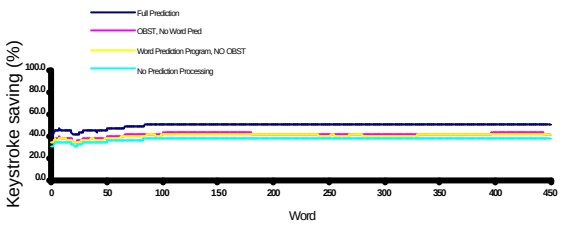



Figure 3. Cumulative keystroke savings for the medical article using various prediction methods.

The suggested structure creates a large data structure of several millions of items. The effect of this problem decreased with the increase in the speed of processors, the increase in storage capacity, and advancements in cluster computing algorithms and machines.

## References

[1] Al-Furaih I., Aluru S., Goil S., and Ranka S., "Parallel Construction of Multidimensional Binary Search Trees," *IEEE Trans. on Parallel and Distributed Systems*, vol. 11, no. 2, pp. 136-148, February 2000.

[2] Brown P., Pietra V., DeSouza P., Lai J., and Mercer R., "Class-Based N-Gram Models of Natural Language," *Computational Linguistics*, vol. 18, no. 4, pp. 467-479, 1992.

[3] Chen S. and Goodman J., "An Empirical Study of Smoothing Techniques for Language Modeling," *in Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, CA, USA, pp. 310-318, June 1996.

[4] Even-Zohar Y. and Roth D., "A Classification Approach to Word Prediction," *in Proceedings of The 1st North American Conference on Computational Linguistics (NAACL' 2000)*, pp. 124-131, 2000.

[5] Garay-Vitoria N. and Gonzalez-Abascal J., "Intelligent Word-Prediction to Enhance Text Input Rate (A Syntactic Analysis-Based Word-Prediction Aid for People with Severe Motor and Speech Disability)," *in Proceedings of International Conference on Intelligent User Interfaces*, January 6-9, Orlando, FL, USA, pp. 241-244, 1997.

[6] Gonnet G., Baeza-Yates R., and Snider T., "New Indices for Text: Pat Trees and Pat Arrays," in Frankes W. and Baeza-Yates R. (Eds), *Information Retrieval: Data Structures and Algorithms*, Prentice Hall, New Jersey, pp. 66-82, 1993.

[7] Heeman P., "POS Tagging versus Classes in Language Modeling," *in Proceedings of the Sixth Workshop on Very Large Corpora*, Montreal, pp. 179-187, August 1998.

[8] Hunnicutt S., "Using Syntactic and Semantic Information in a Word Prediction Aid," *in Proceedings of Europe Conference Speech Commun.*, Paris, France, vol. 1, pp. 191-193, September 1989.

[9] Jelinek F., "Self-Organized Language Modeling for Speech Recognition," *Technical Report*, IBM T. J. Watson Research Center, Continuous Speech Recognition Group, 1985.

[10] Jelinek F. and Mercer R. "Interpolated Estimation of Markov Source Parameters from Sparse Data," *in Proceedings of Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands, North-Holland, pp. 381-397, 1980.

[11] Katz S., "Estimation of Probabilities from Spares Data for the Language Model Component of a Speech Recognizer," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400-401, 1987.

[12] Koester H. and Levine S., "Modeling the Speed of Text Entry With a Word Prediction Interface," *IEEE Trans. on Rehabilitation Engineering*, vol. 2, no. 3, pp. 177-187, September 1994.

[13] Lesher G, Moulton B., and Higginbotham D., "Effects of N-gram Order and Training Text Size on Word Predition," *in Proceedings of (RESNA'99) Annual Conference*, Arlington, VA, pp. 52-54, 1999.

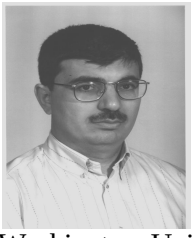
[14] Lesher G., Moulton B., and Higginbotham D., "Techniques for Augmenting Scanning Communication," *Augmentative and Alternative Communication*, vol. 14, no. 2, pp. 81-101, 1998.

[15] Matthew E., "Syntactic Pre-Processing in Single-Word Prediction for Disabled People," *PhD Thesis*, University of Bristol, England, June 1996.

[16] Pereira F., Singer Y., and Tishby N., "Beyond Word N-Grams," *in Proceedings of the Third Workshop on Very Large Corpora*, Columbus, Ohio, Massachusetts Institute of Technology, Association for Computational Linguistics, pp. 95-106, 1995.

[17] Srinivas B., "Complexity of Lexical Descriptions and its Relevance to Partial Parsing," *PhD Thesis*, University of Pennsylvania, IRCS Report 97-10, 1997.

[18] Wessel F., Ortmanns S., and Ney H., "Implementation of Word Based Statistical Language Models," *in Proceedings of SQEL Workshop on Multi-Lingual Information Retrieval Dialogs*, Pilsen, Czech Republic, pp. 55-59, April 1997.



**Eyas El-Qawasmeh** received his BSc degree in computer science in 1985 from Yarmouk University, Jordan. He then joined the Yarmouk University as teaching assistant in the Computer Science Department. In 1992, he joined the George Washington University, Washington DC, USA where he obtained his MSc and PhD degrees in software and systems in 1994 and 1997, respectively. In 2001, he joined George Washington University as visiting researcher. Currently, he is an assistant professor in the Department of the Computer Science and Information Systems at Jordan University of Science and Technology, Jordan. His areas of interest include multimedia databases, information retrieval, and object-oriented. He is a member of the ACM and IEEE.