Unified Perception-Prediction Model for Context Aware Text Recognition on a Heterogeneous Many-Core Platform

Qinru Qiu, Qing Wu, and Richard Linderman

Abstract— **Existing optical character recognition (OCR) software tools can perform text image detection and pattern recognition with fairly high accuracy, however their performance will be significantly impaired when the image of the character is partially blocked or smudged. Such missing information does not hinder the human perception because we predict the missing part based on the word level and sentence level context of the character. In order to mimic the human cognitive behavior, we developed a hybrid cognitive architecture combining two neuromorphic computing models, i.e. brain-state-in-a-box (BSB) and cogent confabulation, to achieve context-aware text recognition. The BSB model performs the character recognition from input image while the confabulation models perform the context-aware prediction based on the word and sentence knowledge bases. The software tool is implemented on an 1824-core computing cluster. Its accuracy and performance are analyzed in the paper.**

I. Introduction

Military planning, battlefield situation awareness, and strategic reasoning rely heavily on the knowledge of the local situation and the understanding of different cultures. A rich source of such knowledge is presented as natural-language text. In 2009, DARPA launched the Machine Reading program to develop a universal text-to-knowledge engine that scavenges digitized text to generate knowledge that can be managed by the artificial intelligence reasoning systems. The Machine Reading program limits its scope to the texts available on the World Wide Web. In real life, text exists in many forms other than its ASCII representation. These include printed texts such as books, newspapers and bulletins or hand written texts. There are many occasions when only the scanned or photographed image of the texts is available for computer processing. While the machine reading system bridges the gap between natural language and artificial intelligence, another bridge has to be constructed to link the natural state of texts to its unique encoding that can be understood by computers.

Conventional Optical Character Recognition (OCR) tools or pattern recognition techniques are not enough to meet the challenges in this task. Because the text images are usually captured under extreme circumstances, sometimes the images will be noisy, or incomplete due to the damages to the printing material, or obscured by marks or stamps. Pattern recognition is extremely difficult, if not impossible, when the image is partially shaded or partially missing. However, such tasks are not too difficult for humans as we predict the missing information based on its context. Most human cognitive processes involve two interleaved steps, perception and prediction. Together, they provide higher accuracy.

Research work in cognitive computing has resulted in many computing models with different mathematical methods and application fields. In one category, computing models have been developed for performing cognitive functions on raw input signals such as image and audio. One representative area in this category is the associative neural network model, which is typically used for pattern recognition. We generally say that this kind of model performs the "perception" function. In the other category, models and algorithms are researched to operate on the concept-level objects, assuming that they have already been "recognized" or extracted from raw inputs. In a recent development, the cogent confabulation model was used for sentence completion [5][6]. Trained using a large amount of literatures, the confabulation algorithm has demonstrated the capability of completing a sentence (given a few starting words) based on conditional probabilities among the words and phrases. We refer these algorithms as the "prediction" models.

In this paper, we present a unified perception-prediction framework that combines the algorithms of neural networks and confabulation. The framework uses neural network models for pattern recognition from raw input signal, and confabulation models for abstract-level recognition and prediction functionalities.

To demonstrate the effectiveness of this framework, we have designed a three-level optical text recognition application targeted at a 1824-core computing cluster (288 6-core nodes and 12 8-core head-nodes) at Air Force Research Laboratory (AFRL). At the lower (character) level, we apply Brain-State-in-a-Box (BSB) neural network models for character recognition from raw image. At the middle (word) level, we have developed a new confabulation model for combining the character recognition results to form words and predict characters. At the higher (sentence) level, another confabulation model is developed to form

Manuscript received February 10, 2011.

Qinru Qiu is with Binghamton University, State University of New York, Binghamton NY 13902 (phone: 607-777-4918, fax: 607-777-4464, email: qqiu@binghamton.edu).

Qing Wu is with USAF AFMC AFRL/RITC, 525 Brooks Road Rome, NY 13441 (phone: 315-330-3219, fax: 315-330-2953, email: Qing.Wu@rl.af.mil).

Richard Linderman is with USAF AFMC AFRL/RITC, 525 Brooks Road Rome, NY 13441 (phone: 315-330-4512, fax: 315-330-2953, email: Richard.Linderman@rl.af.mil).

meaningful sentences and predict words. Experiments are designed to let this framework work on images of scanned text with missing information, i.e., texts with hard-to-recognize or missing characters. Results show that, for texts with 20% distortions randomly distributed on 40% (randomly) of the characters, about 95% of the words are recognized (recovered) correctly and more than 65% of the sentences are recovered correctly.

II. Background

A. Brain-state-in-a-box

The BSB model is an auto-associative, nonlinear, energy minimizing neural network [1][3]. A common application of the BSB model is to recognize a pattern from a given noisy version. BSB model can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries.

There are two main operations in a BSB model, Training and Recall. In this paper, we will focus on the BSB recall operation. The mathematical model of a BSB recall operation can be represented in the following form [4]:

$$\boldsymbol{x}(t+1) = S(\alpha \cdot \boldsymbol{A} \cdot \boldsymbol{x}(t) + \lambda \cdot \boldsymbol{x}(t) + \gamma \cdot \boldsymbol{x}(0))$$

where:

- $\boldsymbol{x}$ is an N dimensional real vector
- $\boldsymbol{A}$ is an NxN connection matrix
- $\boldsymbol{A} \cdot \boldsymbol{x}(t)$ is a matrix-vector multiplication operation
- α is a scalar constant feedback factor
- λ is an inhibition decay constant
- γ is a nonzero constant if there is a need to maintain the input stimulation
- $S(\ )$ is the "squash" function defined as follows:

$$\boldsymbol{S}(\boldsymbol{y}) = \begin{cases} \boldsymbol{1} & \boldsymbol{if\ y \geq 1} \\ \boldsymbol{y} & \boldsymbol{if -1 < y < 1} \\ \boldsymbol{-1} & \boldsymbol{if\ y \leq -1} \end{cases}$$

In [2], we implemented and optimized the recall operation of the BSB model on the Cell Broadband Engine processor. The runtime measure shows that, we have been able to achieve about 70% of the theoretical peak performance of the processor.

B. Cogent Confabulation

Cogent confabulation [5] is an emerging computation model that mimics Hebbian learning, the information storage and interrelation of symbolic concepts, and the recall operations of the brain. Based on the theory, the cognitive information process consists of two steps: learning and recall. During learning, the knowledge links are established and strengthened as symbols are co-activated. During recall, a neuron receives excitations from other activated neurons. A "winner-takes-all" strategy takes place within each lexicon. Only the neurons (in a lexicon) that represent the winning symbol will be activated and the winner neurons will activate other neurons through knowledge links. At the same time, those neurons that did not win in this procedure will be suppressed.

Figure 1 shows an example of lexicons, symbols and knowledge links. The three columns in Figure 1 represent three lexicons for the concept of shape, object and color with each box representing a neuron. Different combinations of neurons represent different symbols. For example, as shown in Figure 1, the pink neurons in lexicon I represent the cylinder shape, the orange and yellow neurons in lexicon II represent a fire extinguisher and a cup, while the red neurons in lexicon III represent the red color. When a cylinder shaped object is perceived, the neurons that represent the concepts "fire extinguisher" and "cup" will be excited. However, if a cylinder shape and a red color are both perceived, the neurons associated with "fire extinguisher" receives more excitation and become activated while the neurons associated with the concept "cup" will be suppressed. At the same time, the neurons associated with "fire extinguisher" will further excite the neurons associated with its corresponding shape and color and eventually make those symbols stand out from other symbols in lexicon I and III.

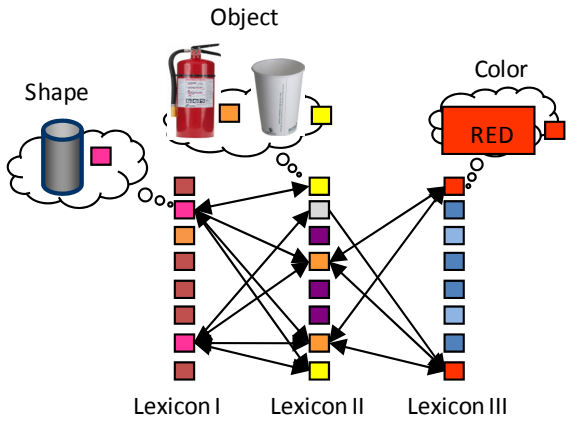

Figure 1. A simple example of lexicons, symbols and knowledge links.

A computational model for cogent confabulation is proposed in [5]. Based on this model, a *lexicon* is a collection of symbols. A *knowledge link* (*KL*) from lexicon A to B is a matrix with the row representing a source symbol in A and the column representing a target symbol in B. The ijth entry of the matrix represents the strength of the synapse between the source symbol s_i and the target symbol t_j. It is quantified as the conditional probability $P(s_i \mid t_j)$. The collection of all knowledge links is called a *knowledge base* (*KB*). The knowledge bases are obtained during the learning procedure. During recall, the excitation level of all symbols in each lexicon is evaluated. Let l denote a lexicon, F_l denote the set of lexicons that have knowledge links going into lexicon l, and S_l denote the set of symbols that belong to lexicon l. The excitation level of a symbol t in lexicon l can be calculated as:

$$I(t) = \sum_{k \in F_l} \sum_{s \in S_k} I(s) \left[\ln\left(\frac{P(s|t)}{p_0}\right) + B \right], \ t \in S_l.$$

The function $I(s)$ is the excitation level of the source symbol s. Due to the "winner-takes-all" policy, the value of $I(s)$ is either "1" or "0". The parameter p_0 is the smallest

meaningful value of $P(s_i \mid t_j)$. The parameter B is a positive global constant called the *bandgap*. The purpose of introducing B in the function is to ensure that a symbol receiving N active knowledge links will always have a higher excitation level than a symbol receiving (N-1) active knowledge links, regardless of the strength of the knowledge links.

III. System Architecture

A. Overview of the ITRS

In this project, we developed the prototype of a context aware Intelligence Text Recognition System (ITRS) that mimics the human information processing procedure. The ITRS system learns from what has been read and, based on the obtained knowledge, it forms anticipations and predicts the next input image (or the missing part of the current image). Such anticipation helps the system to deal with all kinds of noise that may occur during recognition.

The ITRS is divided into 3 layers as shown in Figure 2. The input of the system is the text image. The first layer is character recognition software based on BSB models. It tries to recall the input image with stored image of the English alphabet. In this work, a race model is adopted. The model assumes that the convergence speed of the BSB indicates the similarity between patterns. For a given input image, we consider all patterns that converge within 50 iterations as potential candidates that may match the input image. All potential candidates will be reported as the BSB results. Using the racing model, if there is noise in the image or the image is partially damaged, multiple matching patterns will be found. For example, a horizontal scratch will make the letter "T" look like the letter "F". In this case we have ambiguous information.

The ambiguity can be removed by considering the word level and sentence level context, which is achieved in the second and third layer where word and sentence recognitions are performed using cogent confabulation models. The models fill in the missing characters in a word and missing words in a sentence. The three layers works cooperatively. The BSB layer performs the word recognition and it sends the potential letter candidates to the word level confabulation. The word recognition layer forms possible word candidates based on those letter candidates and sends this information to the sentence recognition layer. There could be feedback paths that send the sentence level confabulation results back to word level or send word confabulation results back to character level. We believe that the feedback information can speed up the recognition process and its implementation will be our future task. Figure 2 shows an example of using the ITRS to read texts that has been smudged. The BSB algorithm recognizes text images with its best effort. The word level confabulation provides all possible words that associate with the recognized characters while the sentence level confabulation finds the combination among those words that gives the most meaningful sentence.

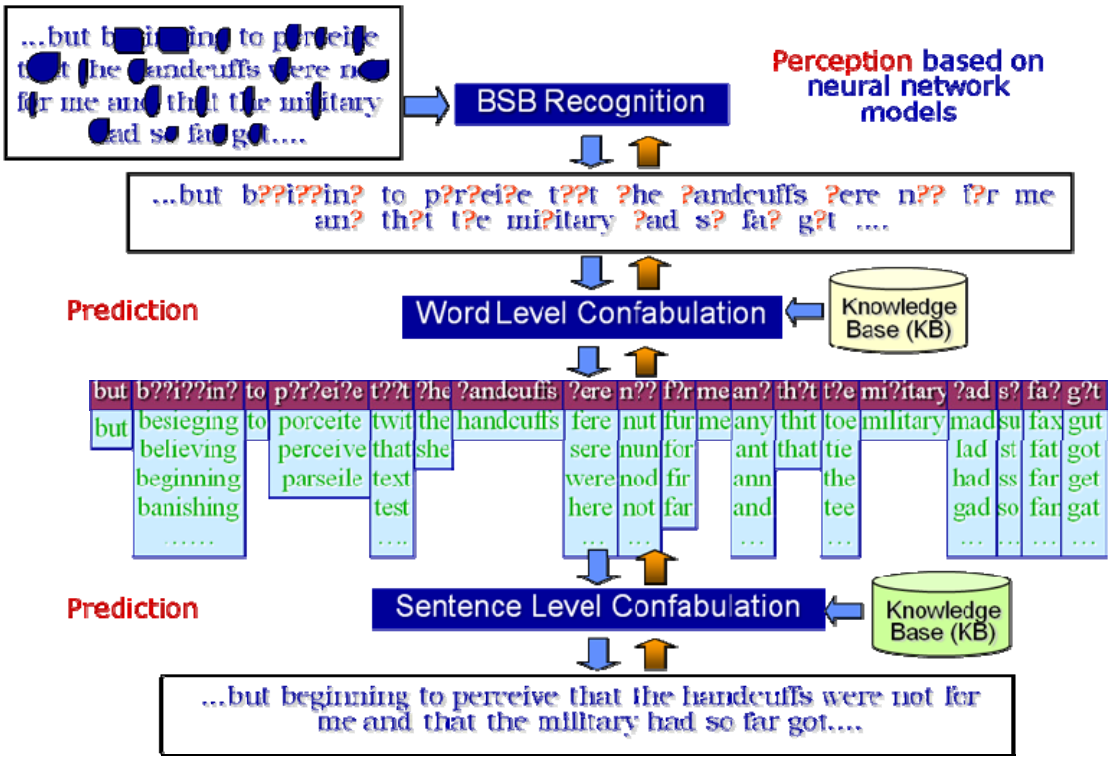

Figure 2. Overall architecture of the models and algorithmic flow.

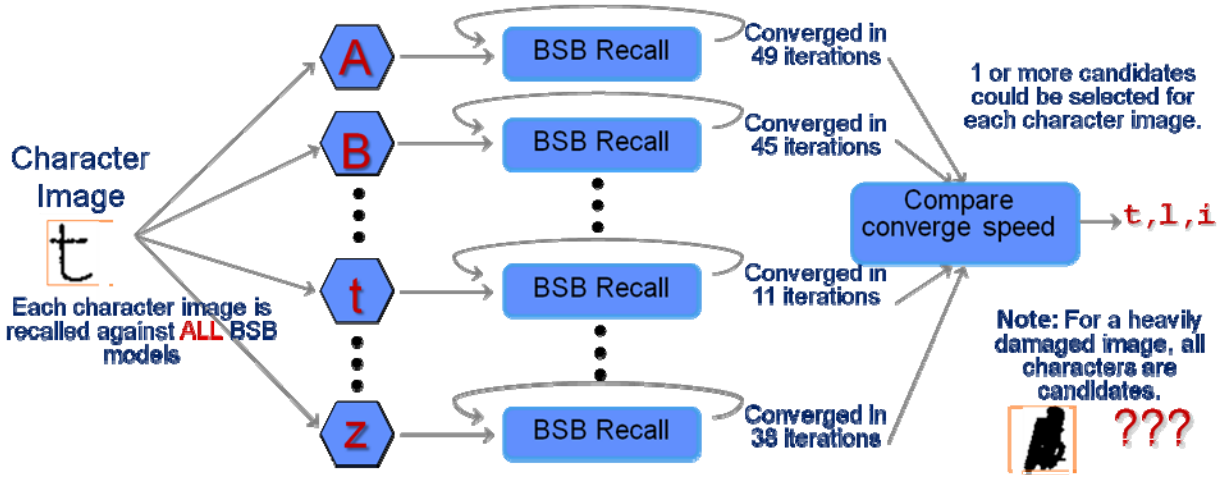

Figure 3. An input image is recalled by BSB models remembering different characters or symbols. Candidates are selected based on speed of convergence.

B. Font End Image Processing

At top level of the ITRS is the image processing function that reads in the image of the text and separates it into blocks of smaller images that contains only one character. The image processing function distinguishes punctuations from texts and uses them to separate sentences. It also separate words based on white spaces. The output of the image processing function is a set of character images labeled by a triplet (i, j, k), where k is the position of the character in a word, j is the position of this word in a sentence, and i is the index of the sentence that the character belongs to.

C. Character Level Image Perception

The output of the image processing function is the input of the BSB based character recognition function. The Brain-State-in-a-Box model is a simple, non-linear, auto-associative neural network. Human memory is said to be associative; that is, one event is linked to another event. Given a vague, partially formed idea or input, an associative memory will compare it against all other stored content until a match is found. In this context, a match refers to the resolution, completion, or connection of the full version, result, or 'answer' based upon the partial input located within memory. The BSB algorithm mimics this auto-associative behavior in that prototype patterns are stored as vectors in the neural network and are recalled when a 'noisy' or incomplete version of the pattern is presented to the system.

As mentioned in the system overview, BSB is used for character recognition within the ITRS. Characters in the system are represented by 15x15 pixel patterns. The system is trained with a character set and when a letter image is presented to the BSB algorithm, it is compared against all models in the system. This comparison is called the 'recall' stage. The 'winning' candidate characters are those that converge, or match, the closest to the images trained in the system. More than one character can be sent to the word-level confabulation algorithm as a candidate, if multiple letters have the same degree of similarity to the input pattern. For particularly damaged characters, all letters in the alphabet can be considered candidates. An illustration of the BSB recall procedures is shown in Figure 3.

D. Confabulation-Based Word Level Prediction

The inputs of word confabulation are characters with ambiguities referred as candidates. For each input image, one or multiple character level candidates will be generated by the BSB model. In this work, we assume that each word has less than 20 characters. Any word that is longer than this will be truncated. If a word has less than 20 characters, it will be padded with white spaces.

The work level confabulation model consists of three levels of lexicon units (LUs). There are 20 LUs in the first level and the ith LU in the first level represents the ith character in the word. There are 19 LUs in the second level and the ith LU in the second level represents a pair of adjacent characters at location i and i+1. Finally, there are 18 LUs in the third level and the ith LU in the third level represents a pair of characters located at i and i+1.

A *knowledge link* (KL) from lexicon A to B is an $M \times N$ matrix, where M and N are the cardinalities of symbol sets S_A and S_B. The ijth entry of the knowledge link gives the conditional probability $P(i|j)$, where $i \in S_A$, and $j \in S_B$. Symbols i and j are referred to as *source symbol* and *target symbol*. Between any two LUs, there is a knowledge link (KL). If we consider the lexicons as vertices and knowledge links as directed edges between the vertices, then they form a complete graph.

Confabulation-based word level and sentence level prediction heavily relies on the quality of the *knowledge base* (KB). The training of the KB is the procedure to construct the probability matrix between source symbols and target symbols. Figure 4 gives a simple algorithm for the construction of the knowledge base. First the program scans through the training corpus and counts the number of co-occurrences of symbols in different lexicons. Then for each symbol pair it calculates their posterior probability.

Reset the co-occurrence matrix $\mathbf{co_{AB}}$ to 0, where $0 \le A, B \le 58$
//count the co-occurrence of symbols
For each sentence in training corpus {
 For each lexicon *A*, $0 \le A \le< 58$ {
  For each lexicon *B*, $0 \le B \le 58$ {
   If $A \ne B$, $co_{AB}[s_A][s_B]$++;
  }
 }
}
//calculate the posterior probability (i.e. knowledge link)
$kl_{AB}[i][j] = \frac{co_{AB}[i][j]}{\sum_i co_{AB}[i][j]}, \forall i \in S_A, \forall j \in S_B, 0 \le A, B \le 39$

Figure 4. A simple algorithm of knowledge base construction.

The word level recall algorithm finds all words from possible combinations of input character candidates. For example, if the input candidates of a 3-letter word are: (w t s r p o k e c a) for the first letter, (h) for the second letter, and (y t s r o m i h e a) for the third letter, then the word level confabulation program will find 24 words, including "why", "who", "wha", "thy", "thi", "the", "tha", "shy", "sho", "she", "rho", "phr", "ohs", "oho", "ohm", "kho", "eht", "cha", "aht", "ahs", "ahr", "ahm", "ahh", and "aha". Note that some of these words are not dictionary words, as it is the nature of a confabulation model to "make up" some new combinations that seem to be reasonable according to its knowledge base.

Figure 5 gives the recall algorithm. For each input candidate in each lexicon, the algorithm sets the corresponding symbols to be active. A lexicon that has multiple symbols activated is referred to as a *ambiguious lexicon* and the goal of the word level confabulation is to eliminate such character level ambiguity as much as possible or to transform it into word level ambiguity which can be further eliminated by sentence level confabulation.

For each lexicon that has multiple symbols activated, we calculate the *excitation level* of each activated symbol. The excitation level of a symbol *i* in lexicon *B* is defined as:

$$EL_B[i] = \sum_{A \ne B} \sum_{j \in \{active\ symbols\ in\ A\}} kl_{AB}[j][i],$$

where $kl_{AB}[j][i]$ is the knowledge value from symbol *j* in lexicon *A* to symbol *i* in lexicon *B*. The *N* highest excited symbols in this lexicon are kept active. These symbols will further excite the symbols in other ambiguous lexicons. This procedure will continue until the activated symbols in all lexicons do not change anymore. If convergence cannot be reached after a given number of iterations, then we will force the procedure to converge.

E. Confabulation-Based Sentence Level Prediction

For each word in a test sentence, the word level confabulation model generates one or multiple word candidates. They will be the input to the sentence level confabulation model.

The sentence level confabulation model is very similar to its word level counterpart except that there are only two levels of LUs. The first level LUs represent single words while the second level LUs represent adjacent word pairs. The training and recall functions of sentence level confabulation have the same principle as these functions at word level. However, it is important to point out that for each word level lexicon there are at most 26 candidates while the number of possible candidates for a sentence level lexicon is countless. This makes the sentence level knowledge base extremely large and to locate an entry in the knowledge base is very time consuming. Two level hash functions are used to speed up the training and recall of the sentence level confabulation model. More details of sentence level confabulation can be found in our recent work [6].

```
For each lexicon and each input candidate of this lexicon
    set the corresponding symbol to be active;
converged = FALSE;
iter = 0;
Set N = MAX_AMBIGUITY;
while(!converged) {
    for each lexicon that has multiple candidates{
        for each candidate i{
            calculate the excitation level of i;
        }
        keep the N highest excited symbols and set the others to be inactive;
    }
    iter ++;
    if (the activation set does not change since the last iteration)
        then converged = TRUE;
    if (iter >= MAX_ITERATION)
        then converged = TRUE;
}
```

Figure 5. Sentence completion: recall.

IV. EXPERIMENTAL RESULTS

A. Experiment Setup

Experiments have been carried out to evaluate the speed and accuracy of proposed ITRS. Four different types of documents, each with different content, lengths, word and sentence-complexities are utilized to obtain a complete understanding of system performance. The first document is an extraction from the novel "Great Expectations" by Charles Dickens; the second document is a piece of business news; the third document is a children's story and the last document is an extraction from a technical paper about speech recognition. As shown in Table 1, among those four documents, the Business and Tech documents have the longest words and the longest sentences in average. This observation justifies some performance trends that will be presented later in this section.

In order to test the robustness of the ITRS, random noises are added to the input text image. We randomly select characters in the test document with probability *p* and modify its image by adding a horizontal scratch that is *s* pixels wide. In our experiment, *p* is varied from 20%, 40% to 60% while *s* is varied from 1, 2, 3, 5 to 9. Because each character image is 15x15 pixels large, increasing thickness of the scratch from 1 to 9 pixels is equivalent to increase the level of distortions from 6.7% (i.e. 1/15) to 60% (i.e. 9/15). A document with 20% of 1-pixel wide scratches is considered "lightly" damaged and a document with 60% of

9-pixel scratches is considered "heavily" damaged. Figure 6 shows some examples of a normal text image and damaged text images.

Table 1. Statistics of Testing Documents.

Document type	#of words	#of sentences	Average word size	Average sentence length
Novel	4012	553	4.0	7.2
Business	955	97	4.6	9.8
Children	1506	221	3.9	6.8
Tech	1063	107	4.3	9.9

Each test document with different scratch severity levels and different levels of scratch probability will be measured for its processing time and accuracy. Time, unless otherwise noted, is the average confabulation time per word or sentence. Accuracy is defined as the percentage of correct character or word identification which results in a correct confabulation. If one character within a letter, or one word within a sentence, is confabulated incorrectly, the entire word or sentence is considered to be inaccurate.

Figure 6. An example of a normal text image and scratched text images.

The training corpus for the word level knowledge base is an English dictionary and the training corpus for the sentence level knowledge base consists of more than 70 classics, including works from Aesop, Louisa May Alcott, James Matthew Barrie, the Bronte sisters, et al.

B. Performance and Accuracy of Word Confabulation (WC)

The first set of measurements shows the average word confabulation (WC) time versus the scratch severity (s) with increasing of scratch probability (p). The results are given in Figure 7 through Figure 10.

A quick glance at the figures will yield three initial observations. The first is that the "Tech" or "Business" documents consistently have the longest average WC times across the entire range of different scratch probabilities for each plot. This is because, as we mentioned before, these two documents has the longest words and sentences. The second observation is that the WC time increases almost exponentially as the scratch probability increases. The third observation is that the difference of the WC time between different documents also increases super-linearly as the scratch probability increases. This means that, as the noise in the image increases, it gets more difficult to read a document that has long words and long sentences.

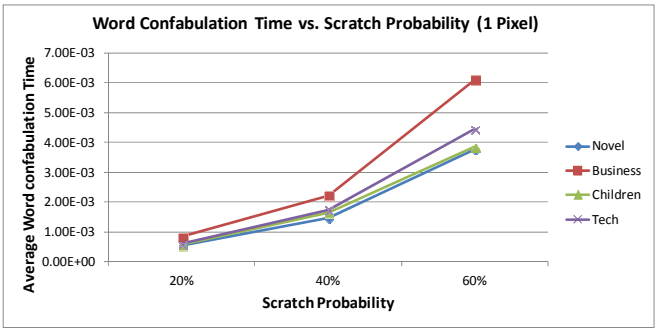

Figure 7. WC time for input files with 1-pixel scratch.

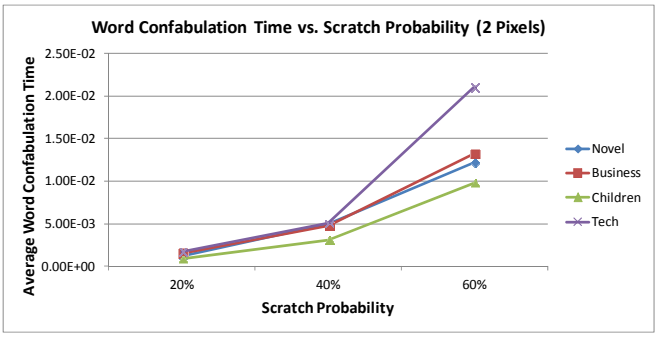

Figure 8. WC time for input files with 2-pixel scratch.

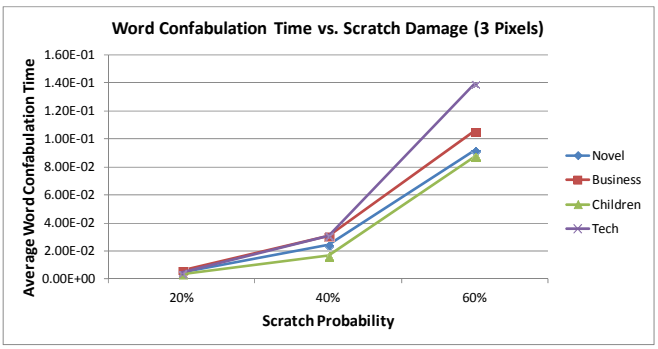

Figure 9. WC time for input files with 3-pixel scratch.

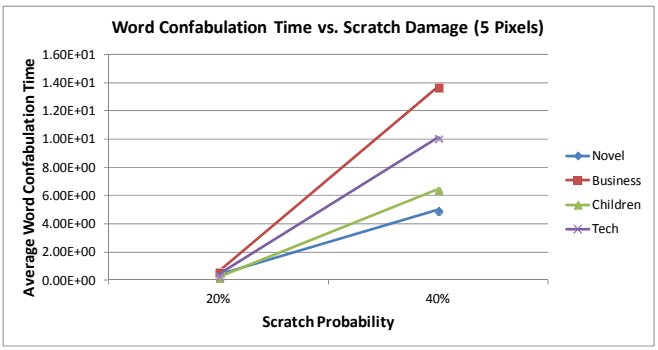

Figure 10. WC time for input files with 5-pixel scratch.

One of the most important performance parameters is the accuracy of the system. Before examining the number of incorrectly confabulated words, it is useful to investigate the number of words that require confabulation as a percentage of the total number of words in a document. Figure 11 shows this information for the test document "Novel" for all scratch probabilities and severities. As we can see, about 4% more words need to be confabulated when the scratch severity

increases from 1 pixel to 9 pixels. This value remains almost constant for all scratch probabilities. This means that when the scratch severity increases, the BSB model can no longer recognize some characters with high confidence as it used to be able to. We also note that, when the scratch probability is 20% document damage, about 57% to 63% of words need to be confabulated; when the scratch probability is 40%, about 74~76% of the words need to be confabulated; and when the scratch probability is 60%, about 81~84% of words need to be confabulated. The percentage of words to be confabulated (i.e. the percentage of ambiguous words) is much larger than the percentage of characters that is scratched out (i.e. the percentage of ambiguous characters). This is because as long as one character in a word is scratched, the meaning of the word cannot be precisely determined.

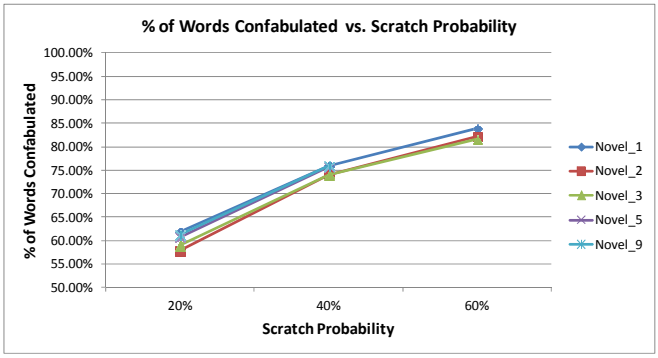

Figure 11. Percentage of words confabulated vs. scratch probability.

Figure 12 shows the number of incorrect word confabulations as a percentage of total word confabulations performed. This is measured for each document at 20% scratch probability for all scratch severities. For scratches sized 1, 2, or 3 pixels wide, the system performs relatively well in terms of accuracy and there is no drastic performance decline until 5-pixel scratches are introduced. At that point, the accuracy of the word confabulations produced by the system begins to degrade rapidly. This trend is present throughout all experiments at any levels of the system.

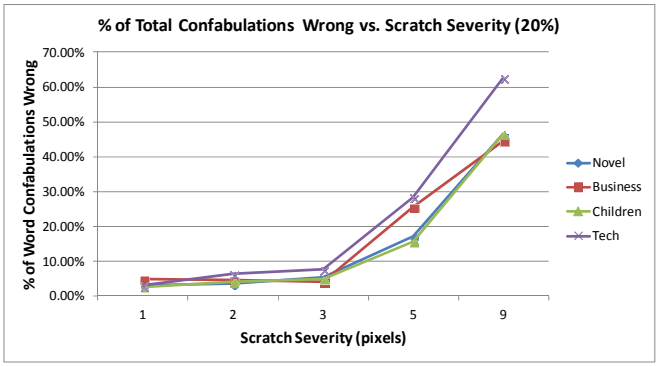

Figure 12. Percentage of incorrect WC at 20% scratch probability.

Figure 13 presents similar results as the previous plot for the input with 40% probability of damage. The amount of inaccurate confabulations produced by WC is still under 10% for scratches under 5 pixels and degrades significantly after that.

In both plots, it is apparent that the test files "Business" and "Tech" cause the system to perform worse than "Novel" and "Children". It is interesting to note that although the average word length of "Tech" is less than that of "Business", the word level inaccuracy of the "Tech" is slightly higher than that of the "Business". This is probably because our sentence level knowledge base is training based a set of classics which are least similar to a science and technical paper.

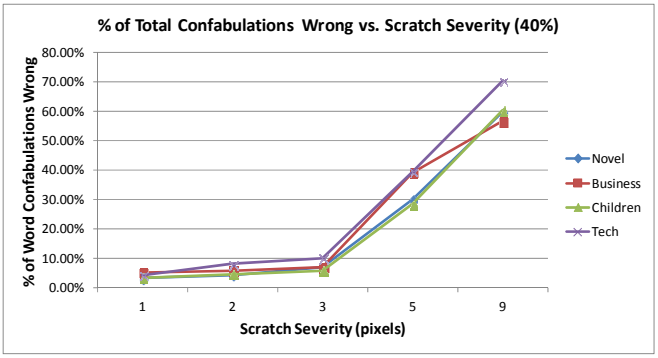

Figure 13. Percentage of incorrect WC at 40% scratch probability.

C. Accuracy of Sentence Confabulation (SC)

Figure 14, Figure 15 and Figure 16 show the sentence recognition accuracy versus the scratch severity for each document type for 20%, 40%, and 60% damage. It is apparent that at 5 pixels, there is an extreme decrease in system accuracy from 1, 2 and 3 pixel scratches. This trend was seen in WC and is certainly present in SC. Again, the children's story book has highest accuracy (i.e. the easiest to read) and the technical paper has lowest accuracy (i.e. the most difficult to read.)

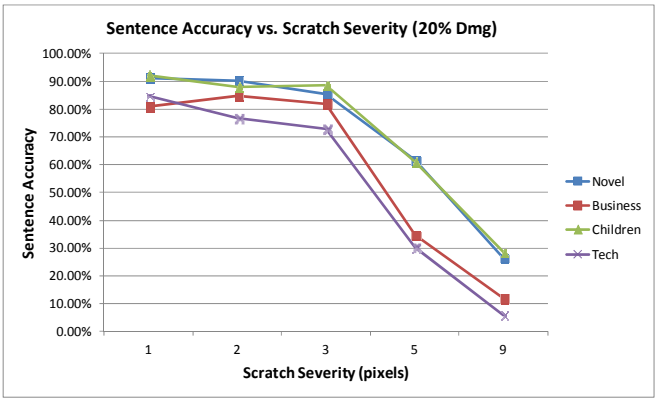

Figure 14. Sentence level accuracy (p = 20%).

At 20% scratch probability, the accuracies for input files that have 1, 2 and 3 pixels scratches are almost constant at 81%, 84%, and 83%, respectively. However, at 5 pixels the accuracy drops to 35%, and at 9 pixels the accuracy is at 13%. That is almost a 50% drop in performance. The gap between a 1, 2, and 3 pixel scratch and a 5 pixel scratch increases slightly when the scratch probability increases. At 40% probability, the difference between 3 and 5 pixel wide scratches is 59%. We cannot get any result for inputs with 5 or more than 5 pixel scratches at 60% scratch probability.

Because there is too much ambiguity and too many candidates, the word and sentence confabulation processes are extremely slow.

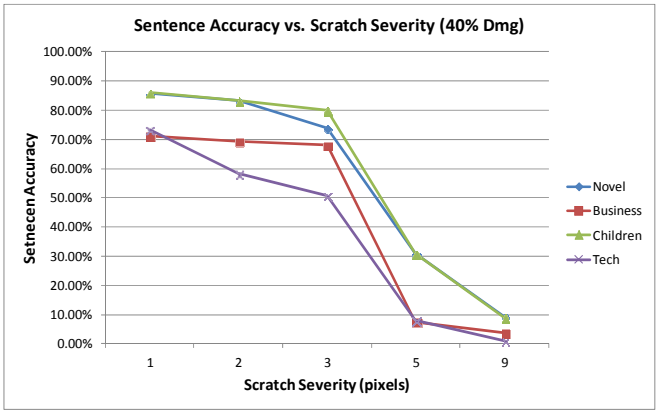

Figure 15. Sentence level accuracy (p = 40%).

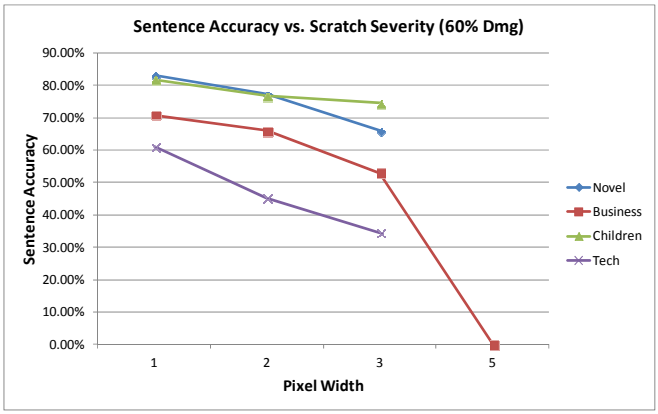

Figure 16. Sentence level accuracy (p = 60%).

Table 2 gives the average sentence level accuracy of all four test documents. It highlights an important observation: the width of the scratch is more damaging to performance than the probability of scratches on a document. That is, the amount of damage to a document is not as destructive as the severity of the damages. As shown by the table, when increasing the scratch probability from 20% to 60%, there is only 9% drop in accuracy if the scratch is 1 pixel wide. Since a 3-pixel scratch represents 20% distortion and a 5-pixel scratch represents 33% distortion in a 15x15 pixel image, increasing the scratch severity from 20% to 33% leads to a 48% accuracy drop at 20% scratch probability.

Table 2. Average sentence level accuracy.

Accuracy	1	2	3	5	9
20%	81%	84%	83%	35%	13%
40%	72%	69%	67%	8%	5%
60%	72%	66%	54%	X	X

D. Overall Runtime

Figure 17 through Figure 19 show the total runtime time to read each test document under different scratch probability and severity conditions. This includes the time for character level, word level and sentence level processing. Note that the overall runtime is proportional to the length of the document. Therefore the overall time to read the test file "Novel" is the longest as it has the most number of words.

In these figures, again, we can see a rapid runtime increase from 3-scratch input to 5-scratch input. This indicates that when the scratch severity increases from 20% to 33%, the ambiguity of the text significantly increases. This does not only impair the reading comprehension of the file and also tremendously increases the reading time.

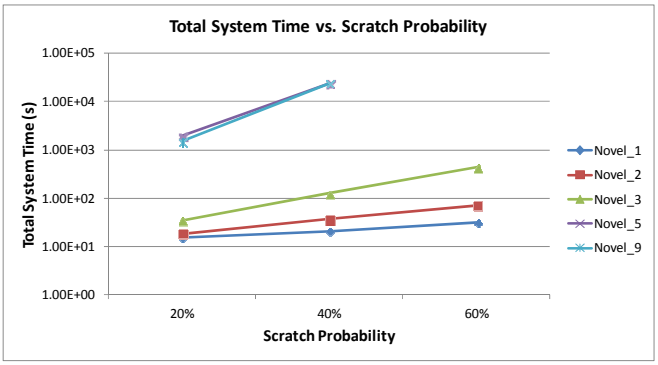

Figure 17. Overall runtime to read test file "Novel".

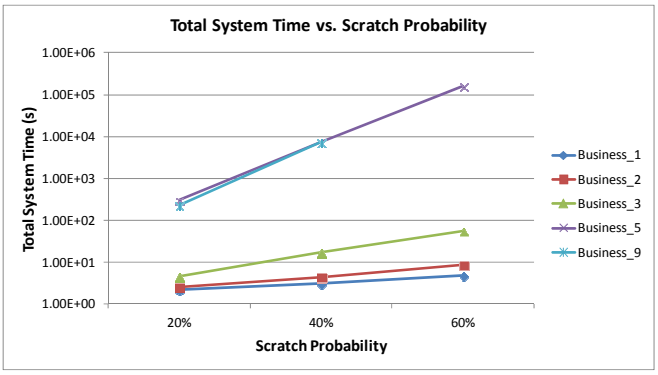

Figure 18. Overall runtime to read test file "Business".

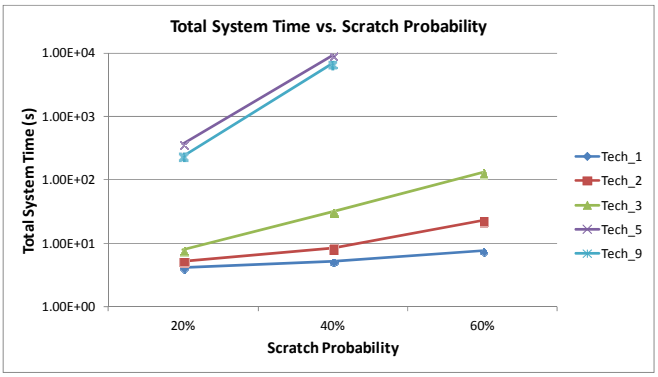

Figure 19. Overall runtime to read test file "Tech".

V. CONCLUSION

In this paper we present a hybrid cognitive architecture combining two neuromorphic computing models, i.e. brain-state-in-a-box (BSB) and cogent confabulation, for context aware text recognition. The BSB model performs the character image recognition while the confabulation models perform the situation-aware prediction based on word and sentence knowledge bases. Its accuracy and performance are analyzed. Our experimental results show that the severity of

the distortion instead of the frequency of occurrence of the distortion has more impact on the accuracy of the software tool.

ACKNOWLEDGMENT OF SUPPORT AND DISCLAIMER

Received and approved for public release by AFRL on 02/07/2011, case number 88ABW-2011-0489.

Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

REFERENCES

[1] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, probability learning: Some applications of a neural model," *in Neurocomputing; Foundations of Research,* J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA: The MIT Press, 1989, ch. 22, pp. 283–325, reprint from Psychological Review 1977, vol. 84, pp. 413–451.

[2] Q. Wu, P. Mukre, R. Linderman, T. Renz, D. Burns, M. Moore and Qinru Qiu, "Performance Optimization for Pattern Recognition using Associative Neural Memory," *Proc. Of 2008 IEEE International Conference on Multimedia & Expo*, June 2008.

[3] "*Associative Neural Memories: Theory and Implementation*," Mohamad H. Hassoun, Editor, Oxford University Press, 1993.

[4] A. Schultz, "Collective recall via the Brain-State-in-a-Box network," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 580–587, July 1993.

[5] R. Hecht-Nielsen, "*Confabulation Theory: The Mechanism of Thought*", *Springer*, Aug. 2007.

[6] Q. Qiu, Q. Wu, D. Burns, M. Moore, M. Bishop, R. Pino, R. Linderman, "Confabulation Based Sentence Completion for Machine Reading," to appear in *Proc. Of IEEE Symposium Series on Computational Intelligence,* April, 2011.