AFRL-RI-RS-TR-2015-063

DOCUMENT IMAGE PARSING AND UNDERSTANDING USING NEUROMORPHIC ARCHITECTURE

SYRACUSE UNIVERSITY

MARCH 2015

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2015-063 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
MORGAN BISHOP
Work Unit Manager

/ S /
MARK H. LINDERMAN
Technical Advisor, Computing
& Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAR 2015	FINAL TECHNICAL REPORT	SEP 2011 – SEP 2014

4. TITLE AND SUBTITLE

DOCUMENT IMAGE PARSING AND UNDERSTANDING USING NEUROMORPHIC ARCHITECTURE

5a. CONTRACT NUMBER
FA8750-11-1-0266

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Qinru Qiu

5d. PROJECT NUMBER
T2MT

5e. TASK NUMBER
0S

5f. WORK UNIT NUMBER
YR

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Syracuse University
900 South Crouse Avenue
Syracuse, NY 13244

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RITB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2015-063

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

In this project, we investigate brain inspired information processing for text image recognition and its performance/accuracy optimization. The intelligent text recognition system (ITRS) works robustly on images with low quality by using a combination of input image data preparation, pattern matching and statistical inference. Our experimental results show that, compared to Tesseract, the ITRS achieves comparable accuracy for clean input images and higher accuracy for camera images with occlusions. Performance enhancement techniques are developed to reduce the processing speed at different layers. In the pattern matching layer, the computing power of multicore processors is explored to reduce the processing time. In the word confabulation layer, new data structures are adopted for the storage of a dictionary, which increases memory locality and reduces search complexity. In the sentence confabulation layer, different sentence models are compared and the best model with the highest accuracy is identified. Finally, the overall ITRS is implemented on a heterogeneous high performance computing cluster. It provides users with the flexibility of computing resource management through a configuration file.

15. SUBJECT TERMS

Neuromorphic computing, brain inspired, text recognition, high performance computing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**MORGAN A. BISHOP**
U	U	U	UU	31	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

Table of Contents

LIST OF FIGURES

1. SUMMARY

In this project, we investigate brain inspired information processing for text image recognition and its performance/accuracy optimization. The intelligent text recognition system (ITRS) works robustly on images with low quality by using a combination of input image data preparation, pattern matching and statistical inference. Our experimental results show that, compared to Tesseract, the ITRS achieves comparable accuracy for clean input images and higher accuracy for camera images with occlusions. Performance enhancement techniques are developed to reduce the processing speed at different layers. In the pattern matching layer, the computing power of multicore processors is explored to reduce the processing time. In the word confabulation layer, new data structures are adopted for the storage of a dictionary, which increases memory locality and reduces search complexity. In the sentence confabulation layer, different sentence models are compared and the best model with the highest accuracy is identified. Finally, the overall ITRS is implemented on a heterogeneous high performance computing cluster. It provides users with the flexibility of computing resource management through a configuration file.

2. INTRODUCTION

With the rapid development in high performance computing (HPC) technologies, the research in machine intelligence has entered a new era. There are many questions to be answered. How to harness the computing power and storage capacity of modern HPC clusters and convert it to useful computations that assist or even replace the human cognition process? Will the performance of current neuromorphic computing models scale as the hardware resource increases? What is the bottleneck of current HPC architectures when applied to cognitive computing and how can this be addressed by future computing tools? This project makes a preliminary effort in answering these questions.

Research discoveries in human psychology suggest that human information processing is a multi-level system [1] that mostly relies on pattern matching and sensory association rather than calculation and logic inference. Information is first processed by the sensory cortex where the complex data is reduced to abstract representations. The abstract representation is compared to stored patterns in massively parallel neural networks in the basal ganglia and neocortex to generate a quick reaction. If more sophisticated processing such as reasoning is needed then relatively slower sequential processes will occur in the prefrontal cortex. To cope with this information process procedure, the neocortex of the human brain consists of the primary sensory area, the association area and the higher order association area [2]. The primary sensory cortex detects the basic dimensions of the external stimuli to the five sensory systems. The sensory cortex is further divided into cortical columns which could detect a specific input pattern (such as contour, color, or pitch, etc.) in a specific area. Each sensory system has its own association area that combines information from the primary sensory cortex to produce perception (i.e. cognition). The higher order sensory system carries out complex mental processes by combining information from several sensory association areas. Sensory association is the most important step in perception. The association area is by far the most developed part of the cerebral cortex.

The above analysis partly reveals the answers to the first question that we previously raised. In order to harness the modern computer to imitate the human cognition process, we believe that the following architecture should be considered:

1. Both the hardware and software should follow the hierarchy of the neocortex system, with the lower level dedicated for pattern detection of the raw external input and the upper level dedicated for information association and statistical inference.

2. The same input should be processed by multiple function modules corresponding to different primary sensory cortical columns for the detection of different patterns. In this way the complexity of each function module is reduced. Furthermore, all of the function modules are independent to each other and can be implemented in parallel.

3. Advanced and fast information association is more important than accurate detection. With the help of information association, relatively simple pattern matching algorithms can be used to achieve accurate perception.

Many algorithms have been proposed for pattern detection and information association. Clearly, different algorithms favor different hardware configurations. In general, pattern matching algorithms such as neural networks and support vector machines (SVM) are dominated by matrix and vector operations while information association models such as Bayesian network and probabilistic graph models require large storage space to capture complex relations. If we try to replicate the human information processing flow, it naturally requires machines with massively parallel processing capability and high computation speed at the bottom layer for pattern detection and machines with large memory space and high memory access speed at the upper layer for inference and information association.

In this project, a proof-of-concept prototype of context-aware Intelligence Text Recognition Software (ITRS) is developed. The lower layer of the ITRS performs pattern matching of the input image using a simple non-linear autoassociative neural network model called Brain-State-in-a-Box (BSB) [3]. It matches the input image with the stored alphabet. Each BSB model is analogous to a cortical column in the primary sensory area that performs the preliminary detection. Sometimes, multiple matching patterns may be found for one input character image. The upper layer of the ITRS performs information association using the cogent confabulation model [5]. It enhances those BSB outputs that have strong correlations in the context of word and sentence and suppresses those BSB outputs that are weakly related. In this way, it selects those characters that form meaningful words and sentences. Each confabulation model is analogous to a cortical column in the sensory association area that associates the primary detections to form high level cognition. Compared to the existing optical character recognition (OCR) system such as OCRopus, Tesseract, and Microsoft OneNote, the proposed ITRS system has the following uniqueness.

1. It has a much simpler bottom layer for image processing and pattern matching. The BSB model is a simple and weak associative memory compared to some more powerful networks using complex learning rules. However, we propose a novel racing mechanism that enables the BSB to generate fuzzy pattern matching results which retain rich information that could be processed by the upper association layer. By contrast, most of the existing text recognition systems heavily rely on image processing and pattern matching, which require complex

algorithms and intensive computation. They provide deterministic results and cannot be integrated with an information association layer.

2. The text recognition of ITRS is mainly achieved by a powerful information association layer. The more than 6GB knowledge base of the information association layer contains information extracted from English literature. It is trained by "reading" more than 70 classical texts. Our hash based technique enables us to update and query the knowledge base efficiently. Although many of the existing text recognition systems also have integrated dictionary and language models, compared to the one used in the ITRS, they are rather primitive.

3. The powerful information association technology and extensive knowledge in English language enables the ITRS to perform text recognition using information beyond the input image. The experimental results show that the ITRS system is capable of recognizing more than 60% of words correctly when each word has 30% characters occluded. Using the OCR function of the Microsoft OneNote, less than 5% of words can be read accurately. While the open source OCR tool Tesseract cannot recognize any of these words with occlusions.

The ITRS explores the computation power of heterogeneous high performance computing clusters. Each cluster node has a host processor based on Intel Xeon architecture, and one (or multiple) co-processor(s) based on NVIDIA GPGPU or Xeon Phi technology. Distributed multithreading streaming process architecture is adopted for the ITRS. The software structure and distribution of computing resource is user configurable.

2.0. Brain Inspired Computing Model

The neuromorphic model adopted by the ITRS software is mainly built based on the *Brain-State-in-a-Box* (BSB) attractor model [3][4] and the *Cogent Confabulation* model [5][6]. The BSB models provide the preprocessing of the image of each character seeking a matching pattern. The cogent confabulation algorithms combine information from the BSB model to form more complex objects such as words or sentences. During this procedure, it suppresses the inputs that do not have strong association with others and enhances the remaining inputs. In other words, the confabulation model eliminates those BSB results that do not form meaningful words and sentences.

2.0.1. BSB model

The BSB model is an auto-associative, nonlinear, energy minimizing neural network. A common application of the BSB model is to recognize a pattern from a given noisy version. It can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries. It has two main operations: *training* and *recall*. The mathematical model of the BSB recall function can be represented as:

$$\mathbf{x}(t+1) = S\left(\alpha \cdot \mathbf{A}\mathbf{x}(t) + \beta \cdot \mathbf{x}(t)\right), \tag{1}$$

where $\mathbf{x}$ is an N dimensional real vector and $\mathbf{A}$ is an N-by-N connection matrix, which is trained using the extended Delta rule. $\mathbf{Ax}(t)$ is a matrix-vector multiplication, which is the main function of the recall operation. α is a scalar constant feedback factor. β is an inhibition decay constant. $S()$ is the "squash" function defined as follows:

$$S(y)=\begin{cases}1, & y\geq 1\\ y, & -1<y<1.\\ -1, & y\leq -1\end{cases} \tag{2}$$

For a given input pattern **x**(0), the recall function computes (1) iteratively until *convergence*, that is, when all entries of **x**(t+1) are either '1' or '–1'.

The most fundamental BSB training algorithm is given in Algorithm 1, which is based on the extended Delta rule. It aims to converge at optimal weights so as to minimize the square of the error between a target output pattern and the input prototype pattern.

Algorithm 1. BSB training algorithm using Delta rule.

Step 0.	Initialize weights (zero or small random values). Initialize learning rate α.
Step 1.	Randomly select one prototype pattern $\gamma^{(k)}\in B^n$, k=1,…,m. B^n is the n-dimension binary space (–1, 1). Set target output to the external input prototype pattern $\gamma^{(k)}$: $t_i=\gamma_i$.
Step 2.	Compute net inputs: $y_{in_i}=\sum_j \gamma_j w_{ji}$ (Each net input is a combination of weighted signals received from all units.)
Step 3.	Each unit determines its activation (output signal): $y_i=S(y_{in_i})=\begin{cases}1, & y_{in_i}\geq 1\\ y_{in_i}, & -1<y_{in_i}<1\\ -1, & y_{in_i}\leq -1\end{cases}$
Step 4.	Update weights: $\Delta w_{ij}=\alpha\cdot(t_j-y_j)\cdot\gamma_i$.
Step 5.	Repeat *Steps 1-4* until the condition $\lvert t(i)-y(i)\rvert<\theta$ is satisfied in m consecutive iterations.

2.0.2. Confabulation model

Cogent confabulation 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.

The confabulation model represents the observation using a set of features. These features construct the basic dimensions that describe the world of applications. Different observed attributes of a feature are referred as *symbols*. The set of symbols used to describe the same feature forms a *lexicon* and the symbols in a lexicon are exclusive to each other. *Knowledge links* (*KL*) are established among lexicons. They are directed edges from the source lexicons to target lexicons. Each knowledge link is associated with a matrix. The *ij*th entry of the matrix gives the conditional probability $\log[p(s_i|t_j)]$ between the symbols s_i in the source lexicon and t_j in the target lexicon. The knowledge matrix is constructed during training by extracting and associating features from the inputs.

The cogent confabulation model has close resemblance to a neural system. The symbols are analogous to neurons and knowledge links between symbols are analogous to synapses between neurons. Whenever an attribute is observed, the corresponding symbol (i.e. neuron) is activated, and an excitation is passed to other symbols (i.e. neurons) through knowledge links (i.e. synapses).

The excitation of a symbol *t* in lexicon *l* is calculated by summing up all incoming knowledge links:

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

where F_l is the set of lexicons that has knowledge links go into lexicon *l*, the function $I(s)$ is the excitation level of the source symbol *s*. 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 their strength. The excitation level of a symbol is its log-likelihood given the observed attributes in other lexicons.

2.1. Performance acceleration using parallel architecture

The target platform for the implementation of the ITRS is a heterogeneous high performance computing cluster. The host processor is a quad-core Intel Xeon processor with simultaneous multithreading (SMT) capability. Two state-of-the-art many-core co-processors are investigated for performance acceleration. The first is the NVIDIA GPGPU and the second is the Intel Xeon Phi processor.

Intel Xeon Phi coprocessor is based on Intel Many Integrated Core (MIC) Architecture. It has more than 50 modified x86 cores integrated on a single die. The x86 compatible architecture allows existing software to be supported with very few modifications, while the multi-core architecture provides huge performance boost. The 7110P Intel Xeon Phi coprocessor used in our experiments has 61 cores with 8GB RAM, each core runs at 1.1GHz and supports 4 hardware threads.

The cores on Intel Xeon Phi coprocessors are simplified in-order execution cores, hence they are small and power efficient. Huge performance boost is obtained by exploiting the massive parallelism and by hiding memory latency. This architecture is a very good example of MIMD

(multiple instruction, multiple data). It supports a total of 244 threads running independent of each other. Thus, fine grained parallelism and good load balancing can be achieved.

There is a 512-bit wide vector processing unit (VPU) on every core providing a Single Instruction Multiple Data (SIMD) capable instruction set. The VPU can execute 16 single-precision or 8 double-precision operations per cycle. It also supports Fused Multiply-Add (FMA) instructions and hence can execute 32 SP or 16 DP floating-point operations per cycle. Vectorization can be assisted by an Intel Compiler or by manually inserting intrinsics and language extensions.

Every core on Xeon Phi has a local L1 and L2 cache. The L2 cache is coherent and its capacity is 512KB. Cache coherence is achieved through a ring interconnect which is 64 bytes wide. Hence each core has the access to a total of 30MB shared L2 cache.

3. METHODS, ASSUMPTIONS, AND PROCEDURES

3.0. Intelligent Text Recognition System

The ITRS is divided into three layers as shown in Figure 1. The input of the system is a text image. The first layer is character recognition software based on BSB models. It tries to recall the input image with stored images of the English alphabet. If there is noise in the image, multiple matching patterns may be found. The ambiguity can be removed by considering the word level and sentence level context, which is achieved by the information association in the second and third layer where words and sentences are formed using cogent confabulation models. Image processing front-end software is designed to read in the scanned images of text and separate them into blocks of smaller images of single characters. The ITRS system is evaluated using images of scanned text with missing information, i.e., texts with hard-to-recognize or missing characters.

In this work, we designed a new “racing” algorithm for BSB recalls. The algorithm is based on the observations that the convergence speed of the BSB recall process indicates the distance between the input image and remembered patterns. For a given input image, we consider all patterns that converge within a certain number of iterations as potential candidates that may match this input image. Candidate BSB outputs will be activated and used to trigger the corresponding symbols in the confabulation model for information association. By using the racing algorithm, if there is noise in the image or the image is partially damaged, multiple matching patterns will be triggered for the same input character image. For example, a horizontal scratch will make the letter “T” look like the letter “F”. In this case we have ambiguity in character recognition. The pattern that cannot form meaningful words and sentences will be eliminated in the later stages.

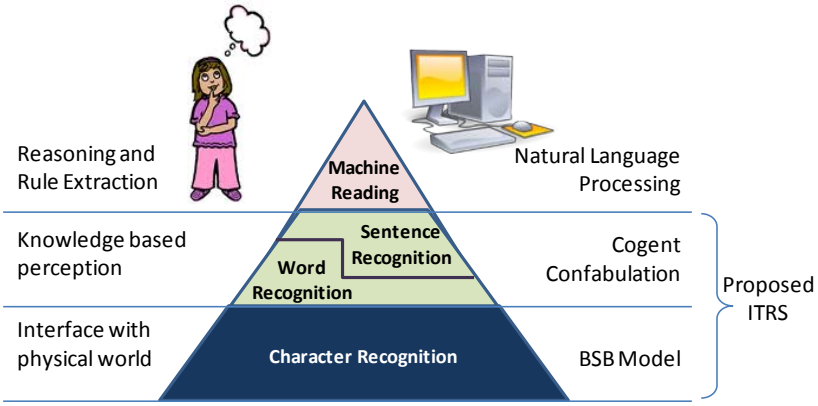


Figure 1 Layered architecture of intelligent text recognition.

Figure 2 shows an example of using the ITRS to read texts that have been occluded. 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. More information about the layered architecture can be found in [10].

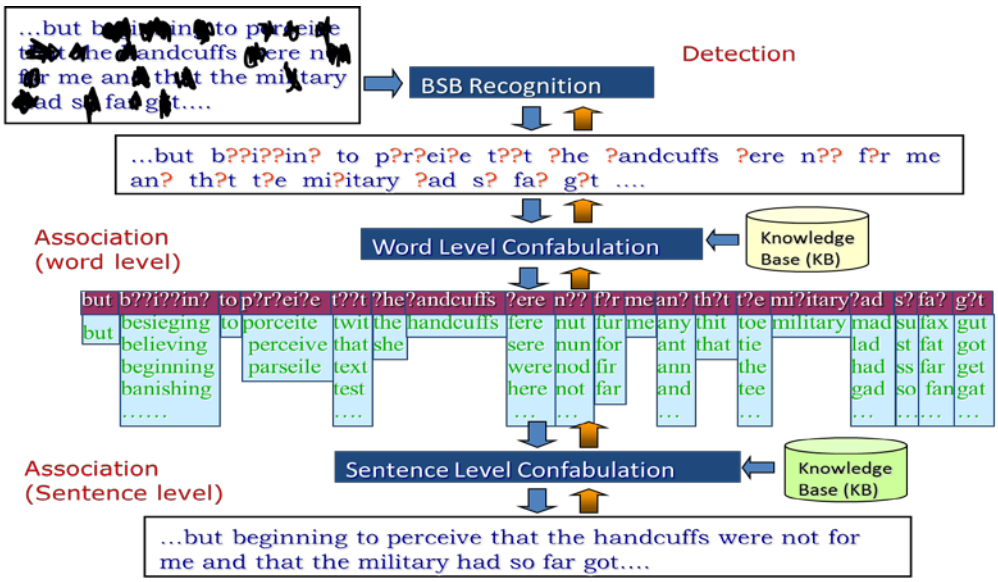
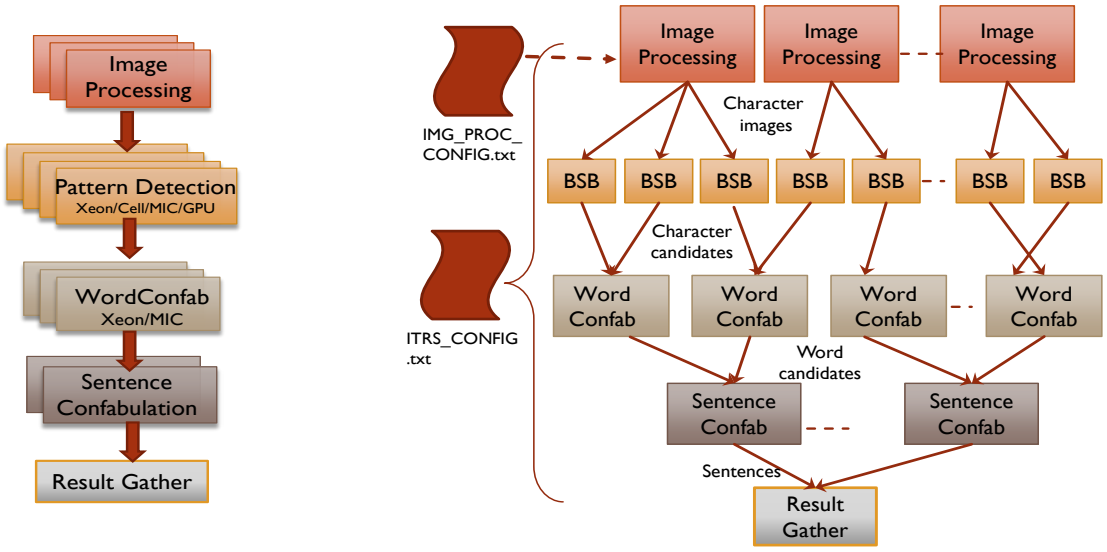


Figure 2 An example of occluded text recognition process.

3.0.1. *Software architecture*

The target computing hardware for ITRS is a heterogeneous high performance computing cluster with state-of-the-art multicore processors such as NVIDIA GPGPU and Intel Xeon Phi and Xeon processors. These processors have different performance. The ITRS should provide the flexibility to maximize resource utilization and maintain a constant throughput in a heterogeneous cluster.

Figure 3 (a) shows the block diagram of the ITRS software architecture. It has 5 layers. In addition to the pattern detection, word confabulation and sentence confabulation layer that were discussed in the previous section, there are also image processing and results gathering which handles input and output functions. Each layer consists of one or multiple MPI modules. Each MPI module consists of one or multiple threads with a dedicated input queue. All threads in the same level employ an identical function but operate on different input data. The image-processing unit breaks the document image into blocks of character images. Each block of character image is a basic workload unit. The workload scheduler assigns workload units to MPI modules in the subsequent layers based on the system configuration file. An example of system configuration is given in Figure 3 (b). Due to heterogeneity of the computing cluster, even in the same level the processing speed of different threads varies. The restructured ITRS provides many control knobs for us to tune the workload and throughput. By setting the number of threads at different levels, specifying CPU affinity of the threads, and controlling the data path, we can optimize the resource utilization and improve the throughput.



(a) (b)

Figure 3 Restructured ITRS (a) workload flow (b) an example of system configuration

3.0.2. *Image processing frontend*

This module handles the job of extracting individual characters from images. It can run as a standalone binary or as part of another module. Once compiled, it can run with various settings provided at runtime. Image processing algorithm has six major steps which need to be performed

in a sequence. To optimize performance these stages are designed as a pipeline as shown in Figure 4. Every stage is designed to run as an independent thread. Each stage can be configured to run with multiple threads. Except for the first stage which always runs with a single thread. For efficient implementation, OpenCV is used to develop this algorithm.

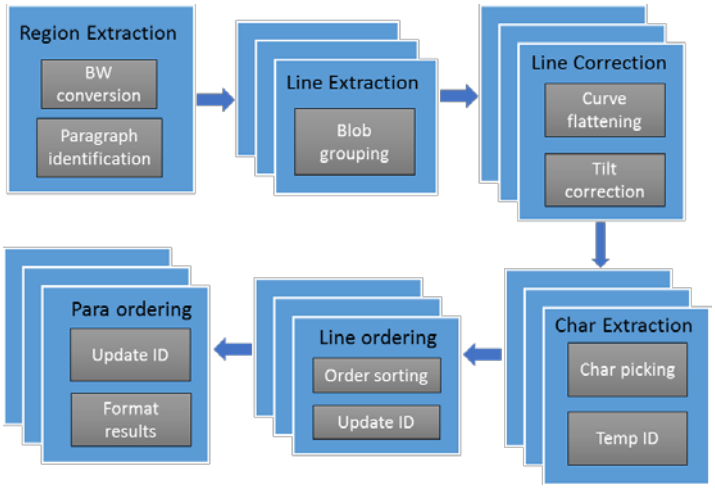


Figure 4 Image processing pipeline

The interface between the pipeline stages is a blocking queue, hence this method of communication is thread safe and also optimizes resource utilization. The functionality of each stage is given below:

Region Extraction This is the first stage in image processing. Region extraction stage operates at the page level. In this stage pages are broken down to paragraphs. It performs basic image filtering to reduce noise in the images and to prep them for Black and White conversion. A 2D gradient of image is computed to extract blob boundaries and removes any illumination gradient usually present if the document image is captured using camera with flash. Every blob is picked up one at a time and converted to black and white using Otsu's method. High fidelity conversion is possible due to dynamic threshold computation for every blob, rather than one threshold for entire image which often leads to poor results.

Morphological erosion is performed on the white background of the image with a rectangular structuring element. The size of the rectangle is chosen such that the height is larger than the text line spacing and width is lower than the column gap in the document image. This operation results in a hole for each paragraph location in the background. Using this as a mask the paragraphs are extracted and labeled. Each paragraph is a workload for the next stage in the pipeline. These workloads can now be processed in parallel to extract text lines from the paragraphs.

Line Extraction This stage operates at paragraph level and the text lines from paragraph are extracted. To begin with, connected components in the paragraph are identified and labeled. The bounding boxes for these blobs are also computed. Now the paragraph image is scanned from top to bottom and left to right. Once a connected component is encountered, it is labeled as an initial blob for the line. The region to the right of the initial blob is scanned to search for neighboring blobs. The width of the search region is large enough to at least include a text space and few

characters of the next word. The height is based on the height of the bounding box of the previous character, in this case the initial blob, plus a few extra rows on the top and bottom for disjoint characters like ‘i’, ‘j’, ‘%’ etc. Any blob in the search region is picked up and marked as current line. The bounding box for the disjoint character is updated to reflect the actual character height. The position of the new blob height is used as starting point for next search. The searching and labeling process continues until the page boundary is reached or until the search space is empty. In this way a text line is labeled and extracted from the paragraph image. Each extracted line is treated as workload for the Line Correction stage.

Line Correction This stage operates at line level. In this stage any deformations due to warping and rotation are corrected. The mid points of all the blobs in the line are used to build a polynomial regression model. The degree of the polynomial is chosen to be 3 as it is sufficient to model the nonlinearities present in the above mentioned distortions. Once the polynomial is computed, the values of the coefficients are analyzed and the degree of the polynomial is reduced to avoid over estimation. The polynomial curve is then used as a reference line that is scanned from top to bottom of the extracted line. All pixels along the scan curve are saved as a flat line thus eliminating warping. At this stage the characters will be tilted if there was rotation. The y-offset of the starting point and the end point of the reference line are computed. This offset is the indication of angle of rotation as shown in Figure 5 (left). The angle θ is computed using the relation $\theta = \tan^{-1}\left(\frac{y_{offset}}{Line\ length}\right)$. Figure 5 (right) represents the flattened scan lines with the gray region representing the actual text region and solid horizontal lines representing each pixel row. The tilt can now be corrected by offsetting the starting points (P1, P2 … Pn) of these lines. The angle ϕ is computed using the relation $\phi = \frac{\pi}{2} - \theta$ and the shift distance for each point is given by the equation $x_{offset} = \frac{row}{\tan(\phi)}$. This operation corrects the tilt introduced due to rotation. If the line is tilted with an angle "$-\theta$" then the points are shifted in the opposite direction. The corrected line forms the workload for character extraction stage.

Figure 5 (left) Angle computation (right) Tilt correction

Character Extraction This stage operates at the line level. Here the characters are extracted and scaled resulting in removal of perspective distortion. The first step is to morphologically group connected component blobs into individual characters. These are extracted and labeled with page ID, paragraph ID, sentence ID, word ID and character ID. The sentence ID is incremented when the image of symbols ‘,’ or ‘.’ is encountered. These two symbols are identified based on the dimensions and position statistics of the character image in the line. The word ID is incremented when a space beyond the threshold is detected between the characters. This stage also splits any connected characters or occlusions based on the configuration specified. Once the characters are labeled they are scaled to either 15x15 or 30x30 resolution. This scaling to a fixed size eliminates perspective distortions introduced in camera captured images. Each

paragraph workload from Region Extraction stage is assigned a blocking counter called Line Counter. After character extraction is finished it increments this counter. Hence this counter keeps track of the number of lines processed in the paragraph. If all lines are processed then the parent paragraph of the current line is scheduled to Line Ordering stage.

Line Ordering This stage operates at the paragraph level. This stage determines the correct order of text lines in the paragraph region. To begin with, it computes the average angle of tilt of all the lines in the paragraph. This information gives the angle of rotation of the image (here paragraph). The starting point of each text line is rotated back by the angle computed. The rotation is performed using the following equations:

$$\begin{bmatrix} RP_{x1} RP_{x1} & & RP_{xn} \\ RP_{y1} RP_{y2} & \cdots & RP_{yn} \end{bmatrix} = H * \begin{bmatrix} LP_{x1} LP_{x1} & & LP_{xn} \\ LP_{y1} LP_{y2} & \cdots & LP_{yn} \end{bmatrix}$$

Where H = is the rotation matrix $\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$

$$\theta = \frac{\pi}{2} - average\ angle$$

LP = Line point

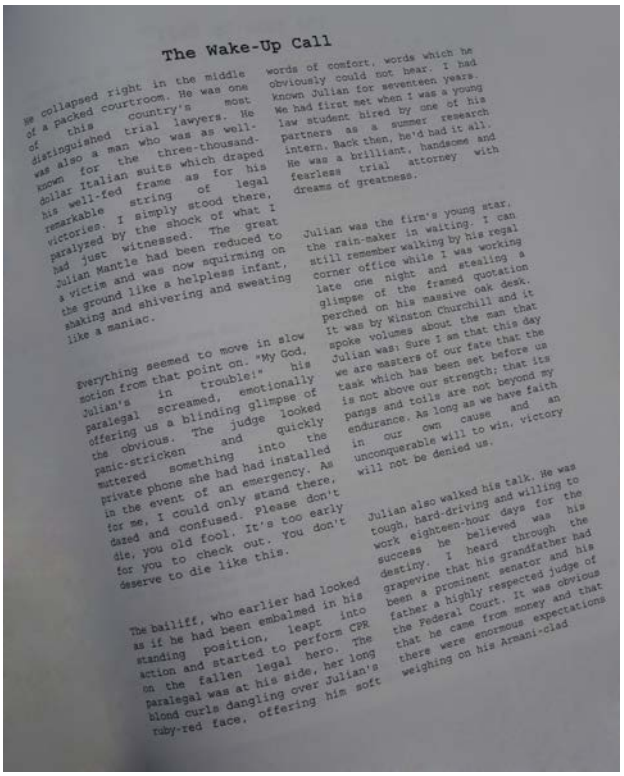
RP = Rotated point

By doing this the y co-ordinates of these points correctly represent the line order. This process has now sorted the lines in order. Using this information the sentence ID and word ID of all the extracted characters in this paragraph are updated. The result at the end of this stage reflects correct sentence ID, word ID and character ID at the paragraph level. After all paragraphs are processed then the parent page of the current paragraph is scheduled to Para Ordering stage.

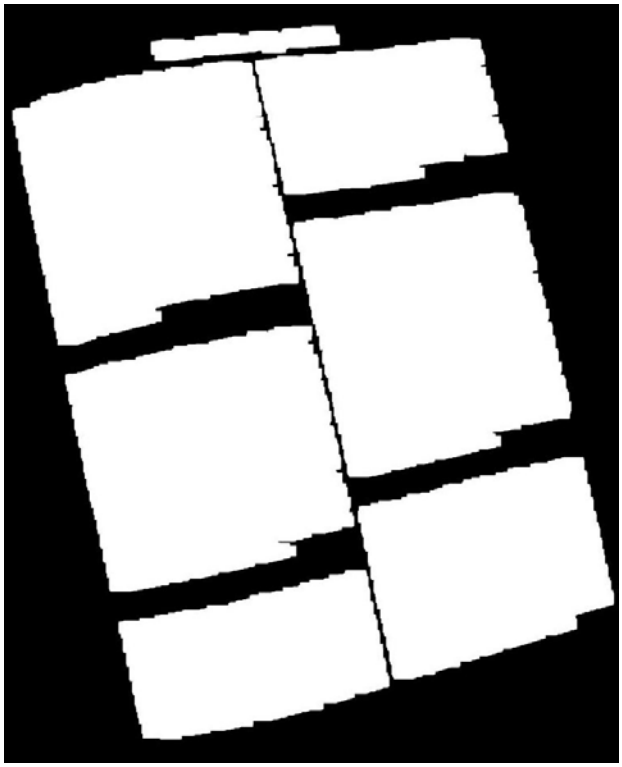
Paragraph Ordering This stage operates at the page level. In this stage paragraph ordering and result generation is performed. All the paragraph starting points are rotated in a similar way as in the Line Ordering stage. These starting points are sorted based on column position. The first paragraph starting position is used as reference and any paragraph whose starting position falls within a threshold of column positions is grouped as one column. This process is repeated till all the paragraphs are grouped into columns. After this grouping each column group is sorted row wise to get the proper order. Based on this order the sentence ID and word ID of all the extracted characters are updated to reflect the correct order.

After ordering is completed the results are saved in a vector format for use by other modules and the results can be saved JPEG images or as training format compatible with pattern recognition module (BSB).

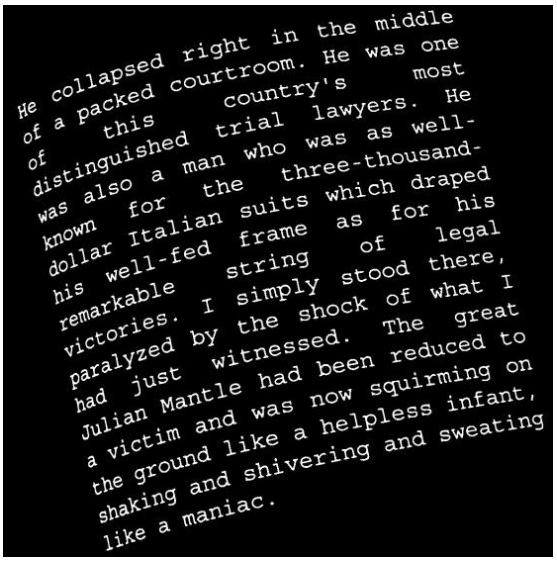
Figure 6 shows an example of an input document image and the intermediate result image after each image processing step.



(a) Original page



(b) Page image after morphological erosion



(c) Intermediate image after region

(d) Intermediate image after line extraction

(e) Intermediate image after line correction

Figure 6 Intermediate results during image processing

3.0.3. Pattern recognition layer

The pattern recognition layer is based on BSB attractor model. In this section we describe the "racing" mechanism that we use to implement the multi-answer character recognition process.

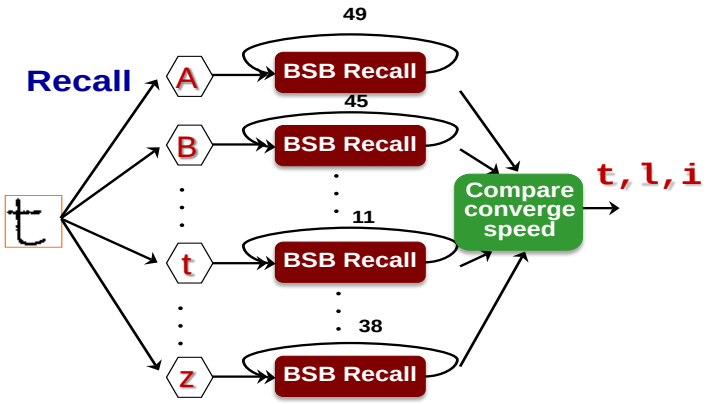


Figure 7 Candidates are selected based on speed of convergence.

Let *S* denote the set of characters that we want to recognize. Without loss of generality, assume the size of *S* is 52, which is the number of upper and lower case characters in the English alphabet. We also assume that for each character, there are *M* typical variations in terms of different fonts, styles and sizes. In terms of pattern recognition, there is a total of 52**M* patterns to remember during training and to recognize during recall.

One 256-dimensional BSB model is trained for each character in *S*. Therefore there will be a set of 52 BSB models. Each BSB model is trained for all variations of a character. The multi-answer implementation utilizes the BSB model's convergence speed to represent the similarity between an input image and the stored pattern. An input image is compared against each one of the 52 BSB models; therefore it triggers 52 recall processes. The number of iterations that each recall process takes to converge is recorded. Then we pick up to *K* "closest" candidates to work with high-level language models to determine the final output. Figure 7 gives an example of how the racing mechanism works.

3.0.4. Word confabulation layer

This module interfaces between BSB and Sentence confabulation. Its role is to collect ambiguous character inputs from BSB layer and generate valid combinations to form meaningful words. The architecture of word confab is shown in the left of Figure 8. The MPI communication module is integrated with word confabulation thread for MPI communication from-BSB and to-Sentence confabulation modules. All inter thread communication is achieved through the thread safe blocking queue. A word scheduler thread communicates with the MPI Communicator, and groups the results into words. These words are scheduled to Confabulation threads using the Scheduling queue. The number of confabulation threads created is user specified.

The scheduler thread uses a map data structure to keep track of what all characters it has received known as word pool. From every received character a unique word level string key is built. The key is the concatenation of [Page-ID]_[Sentence-ID]_[Word-ID]. The word pool consists of word lexicons as its data members. If a key is used for the first time then a new word lexicon is created and the received character candidates are added to it. If a word lexicon already exists then these letter candidates are added to the existing one. After adding the candidates it is checked if all candidates have been received for that word lexicon. After receiving all candidates, the word lexicon is placed on the Scheduling queue and the map entry is erased.

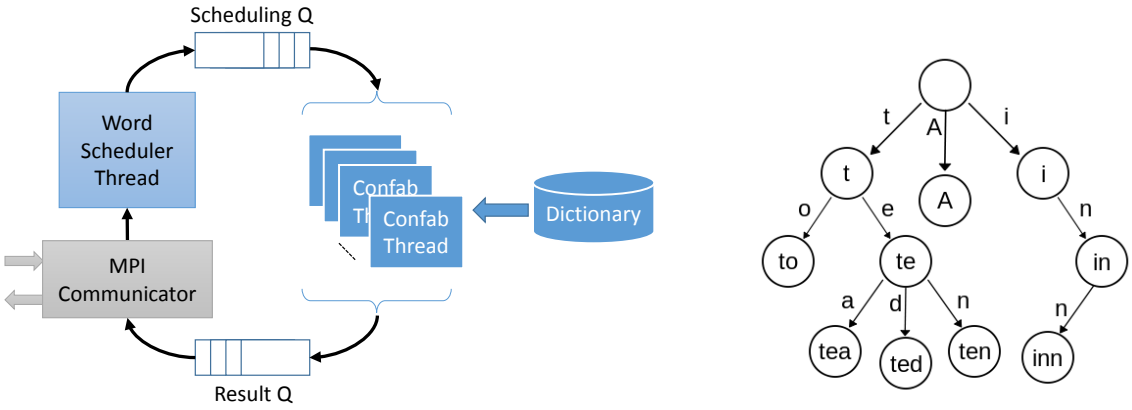


Figure 8 (Left) Word Confabulation architecture (Right) Trie data structure

The confabulation threads use the ambiguous letter candidates and create valid word combinations. The dictionary database is loaded as a trie data structure during initialization of module. This trie is shared between all threads which read this data structure to validate word combinations. An example of trie data structure is shown in the right of Figure 8. Each node holds following members

- Address to parent
- Address of children
- Content
- Word Marker

The combinations based on letter candidates are validated against the trie. For example let's consider a word "dog". Its candidates for each letter position are [d o b] [o a g] [g a y]. Word confab will traverse through the trie using these candidates to search for the valid words present in the trie. The valid words will be pushed onto a stack. In this example, these valid words would be:

- Dog, day, boy, bag.

Since the letter candidates were passed with their relative confidence level, the confidence level for each word will be the product of the letters it contains.

These results are converted to character message. If the size of this message is large than the configured communication buffer, then they are split into multiple messages. These are now sent to sentence confabulation module.

3.0.5. Sentence confabulation layer

The sentence confabulation layer enhances those word candidates that have strong correlations in the context of sentence and suppresses words that are weakly related. In addition to word level and phrase level probability, our model also considers semantic and syntactic properties of sentences by integrating *parts-of-speech* (*POS*) tagging with sentence confabulation [11]. POS tagging [8][9] is a matured technique developed for natural language processing. One of the most widely used probabilistic tagging systems is the Stanford POS Tagger [10]. It is based on the 36 word level tags specified by the Penn Treebank Tagging system. Table 1 lists some examples of these tags.

Table 1 Examples of Penn Treebank Tags

Tag	Function	Example
CC	Coordinating conjunction	and, or, but, …
CD	Cardinal number	one, two, three, …
DT	Determiner	the, this, any, …
EX	Existential there	there, ...
IN	Preposition or subordinating conjunction	of, for, with, …
JJ	Adjective	worthy, clean, sick, …
NN	Noun, singular or mass	kettle, curiosity, …
NNS	Noun, plural	infants, noses, …
VB	Verb, base form	tell, eat, …
VBD	Verb, past tense	told, began, …

We assume that the maximum length of a sentence is 20 words. Any sentence that is longer than 20 words will be truncated. Any sentence that is shorter than 20 words will be padded with empty spaces. The model consists of three levels of lexicons. Lexicons 0 through 19 belong to the first level. Each level 1 lexicon associates to a single word in the sentence. The *i*th lexicon represents the *i*th word. Lexicons 20~38 belong to the second level. Each level 2 lexicon associates to a pair of adjacent words. The lexicon labeled (20+*i*) represents the pair of words in the (*i*+1)th and (*i*+2)th location. Lexicon 39~58 are the POS tags for word lexicons 0~19. Associated to each lexicon is a collection of symbols. A symbol is a word, a pair of words or the POS tag that appears in the corresponding location. We use S_A to denote the set of symbols associated to lexicon *A*.

A knowledge link (KL) from lexicon A to B is a $M \times N$ matrix, where M and N are the cardinalities of symbol sets S_A and S_B. The *i,j*th 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 as *source symbol* and

target symbol. To reduce complexity, knowledge links are established only between lexicons whose horizontal distance is within -5 and +5. Furthermore, KLs are merged into one matrix if their source and target have the same relative position in sentence. For example, the distance from lexicon 0 to lexicon 1 is the same as the distance from lexicon 1 to lexicon 2, so the KL matrices between lexicons 0~1 and lexicons 1~2 are merged.

During training, the reference text is passed through Stanford POS tagger first to generate their respective tags. Knowledge links are established between word lexicons and tag lexicons, but not between word-pair lexicons and tag lexicons. This is because the word pair knowledge links are derivatives of the word knowledge links; therefore they are not needed to build knowledge links with tags.

Since a sentence without tags is given for testing, the confabulation model automatically assumes all tags are possible candidates for all tag lexicons. The system then calculates the excitation level for all the possible tag candidates and eliminates the least excited tag. This process repeats until a single tag is chosen. This elimination method allows multiple candidates to compete throughout the confabulation process and provides more cognitive capacity.

Figure 9 shows an example of a given training sentence and its corresponding lexicon structure. The sentence is "the treasure may be hard to find". The tags are:

```
the_DT treasure_NN may_MD be_VB hard_JJ to_TO find_VB
```

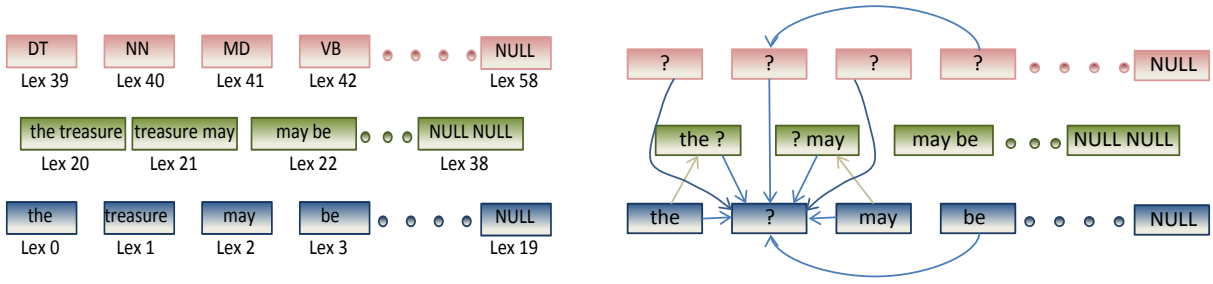


Figure 9 Tag-assisted Sentence Confabulation (a) training (b) recall

In order to extend it to 20 words, we pad 14 empty words and tags to the end of the sentence. Each word, word pair and tag will be entered into lexicons. The system will update the value of all knowledge links. For example, the *KL* from lexicon 0 to lexicon 1 will be adjusted by increasing the conditional probability $P("the"|"treasure")$. The *KL* from lexicon 0 and lexicon 39 will also be adjusted by increasing the conditional probability $P("the"|"DT")$. Obviously, if the words and tags have frequent co-occurrence, their corresponding entry in the knowledge link will have a high value.

Once all training texts are processed, the training process is complete and all final knowledge links are available for the Recall function. Figure 9 is a very simple illustration of the recall function that uses the confabulation model to complete a test sentence with an unknown word and tags. For the sake of illustration, the testing sentence is the same as the training sentence,

with the word “treasure” missing and without pre-processed POS tags. Each square still represent lexicons at different levels and question marks indicate pieces of missing information.

The original sentence confabulation model considers all word candidate inputs equal possibility. However, in reality, we can tell that some words a more likely than others from the given image. To incorporate the image information with confabulation, we integrate the BSB convergence speed into the confabulation process. The new confabulation process calculates the excitation level of word candidate *t* as:

$$I(t) = \alpha P_{BSB}(t) + \beta[\sum_S I(s)P(s|t) + NB], \quad (3)$$

where $P_{BSB}(t)$ is the excitation to *t* from the BSB layer. It is calculated as: $P_{BSB}(t) = \frac{1/(N_{BSB}(t)-N_{min})}{\sum_t 1/(N_{BSB}(t)-N_{min})}$, where $N_{BSB}(t)$ is the BSB convergence speed of *t*, N_{min} is the minimum convergence number that is possible for BSB engines, α and β are coefficients that adjust the weight of BSB (i.e. image) information and confabulation (i.e. language) information, $\alpha + \beta = 1$. In general, we should increase the value of α and decrease the value of β when the image has high quality and vice versa.

We further assign confidence level to the words recognized by ITRS. The confidence level is calculated as the normalized excitation difference between the selected candidate and its final competitor in the last round of confabulation, $c(t_1) = \min[1, \frac{I(t_1)-I(t_2)}{I(t_2)}]$, where t_1 is the selected word and t_2 is its only competitor in the last round of confabulation. For all correct words, on average their excitation level is approximately 2.1 times of the excitation level of their final competitors.

3.1. Extending sentence confabulation to Chinese language

We further extend the ITRS to process documents in Chinese language. The main difference between the Chinese ITRS and the English ITRS is a new pattern detection layer, which recognizes Chinese characters, and a new sentence confabulation layer, which captures and recovers linguistic information of Chinese.

Each Chinese character, which is represented as a 3-byte UTF-8 code, is analogy to an English word. They occupy the word level lexicons in the confabulation framework. Modern Chinese language is based on word compound, which consists of 1~4 single Chinese characters. These word compounds are not delimited, however, they can be found with the help of tools, such as the Stanford POS tagger. We label each Chinese character based on its position in a word compound, and refer this as *segmentation label*. For example, in a two character word compound 书籍(book), 书(book) is located at the first position of the two character word compound, therefore, it is marked as 1IN2, and 籍(book) is marked as 2IN2. In this work, ten segmentation labels are used. They are: 1IN1, 1IN2, 2IN2, 1IN3, 2IN3, 3IN3, 1IN4, 2IN4, 3IN4, and 4IN4. Please note that segmentation label is only needed in Chinese sentence confabulation. This is a major difference between Chinese and western languages.

In the improved confabulation model, new lexicons are created for tags and segmentation labels. Therefore, lexicons in the new confabulation model can be divided into four *levels*. Moreover, instead of having lexicons for two adjacent words, we create lexicons for three adjacent words in order to adapt to semantic compounds of multiple Chinese characters.

Segmented and tagged training text is used during training. Characters, tags and segment labels are placed in corresponding lexicons. KLs are established not only between two lexicons in the same level, but also between lexicons in different levels, as long as their distance is less than 5. However, there is no KL between tag and segmentation label lexicons, because tags and segments are derivatives of the Chinese characters, and Stanford tools are not able to ensure 100% accuracy in tagging. Keeping KL between tag and segment lexicons will introduce noise in the confabulation procedure.

Other than the difference in the lexicon definition and knowledge link connections, the rest of training and recall procedure for Chinese sentence confabulation is exactly the same as the English sentence confabulation.

In the basic confabulation model, the excitation level of a candidate is the sum of contributions from active symbols in other lexicons. Intuitively, however, different source lexicons do not contribute equally to a target lexicon. For example, the lexicon right next to an unknown word obviously gives more information in determining the unknown word than the lexicon that is five words away. This motivates us to weight KL's contribution during recall.

The basic idea is to weight the contribution of each KL based on the *Mutual information* (*MI*) between its source and target lexicons. Mutual information of two random variables is a measure of variables' mutual independence. In our work, mutual information is calculated as

$$I(A;B) = \sum_{b\in B}\sum_{a\in A} p(a,b) log\left(\frac{p(a,b)}{p(a)p(b)}\right),$$

where A is the source lexicon and a represents symbols in A; B is the target lexicon and b represents symbols in B. $p(a,b)$ is the joint probability of symbol a and b; $p(a)$ and $p(b)$ are the margin probability of symbol a and b respectively. $I(A;B)$ is nonnegative. The value of $I(A;B)$ will increase when the correlation of symbols in lexicon A and B get stronger. Because each knowledge link has its source and target lexicons, in the rest of the paper when we say the MI of a KL we refer to the MI of the source and target lexicons of that KL. Figure 10 shows the mutual information of all knowledge links. Based on their connections, the KLs are divided into 9 groups. The group division is described in Table 2 and labeled in Figure 10 underneath the X-axis.

Table 2 Knowledge link group division

KL group	KL IDs	Connection
(a)	0~9	Between word lexicons
(b)	10~19	Between word triplet lexicons
(c)	20~29	Between tag lexicons
(d)	30~39	Between segmentation label lexicons
(e)	40~61	Between word and word triplet lexicons
(f)	62~83	Between word and tag lexicons
(g)	84~105	Between word and segmantation lavel lexicons
(h)	106~127	Between word triplet and tag lexicons
(i)	128~149	Between word triplet and segmentation label lexicons

We can see from Figure 10 that, from left to right, the MI of the KLs in the same group are clustered together. And as the distance between the source and target lexicon of the KL increases, the MI of the KL decreases. For example, KL0 and KL8 belong to the same group; therefore, they have similar MI. However, since KL0 connects between two immediate neighboring lexicons while KL8 connects between two word lexicons that are 4 words apart from each other, the MI of KL0 is slightly greater than KL8. This agrees with our intuitions that adjacent characters have stronger correlations. Second, the KLs connecting to word triplet lexicons always give more information than others; therefore they should be weighed as the biggest during the recall. Based on the above analysis, we assign the weight of a KL as a linear function of its mutual information as shown in the right of Figure 10.

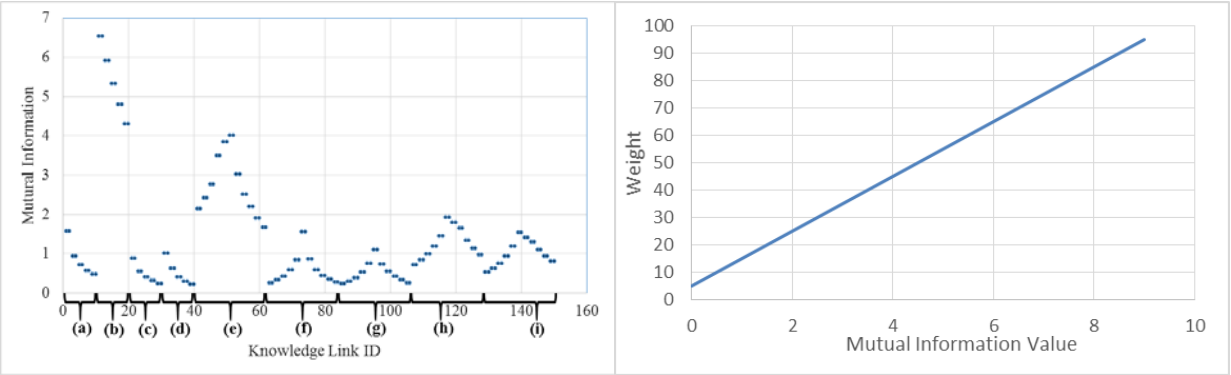


Figure 10 (Left) Knowledge Links' mutual information (Right) Knowledge Links' weight

4. RESULTS AND DISCUSSIONS

We have continuously been improving the performance and accuracy of each layer in the ITRS. Experiments have been carried out to evaluate the impact of our improvements.

4.0. Performance improvements on pattern matching layer

In order to evaluate the impact of different multicore architecture on the performance of the bottom layer of ITRS (i.e. the BSB pattern matching layer), the same pattern-matching layer is implemented on four different processors, Intel Xeon processor, Intel Xeon Phi process, IBM Cell processor and the NVIDIA GPGPU processor. Every time a set of 96 character images is sent to a BSB engine and compared with 93 stored patterns. Each comparison involves 50 iterations of calculation of Equation (1). Hence, $93 \times 96 \times 50 = 446{,}400$ BSB iteration is considered as one unit workload in the BSB layer. In order to represent a 15x15 pixel character image, the input vector $\boldsymbol{x}(t)$ of the BSB is set to 256x1 dimension and the weight matrix $\boldsymbol{A}$ is 256x256. Overall, each unit of BSB workload consists of 58 billion floating point operations. Each stored pattern is handled independently by a thread. In total each BSB workload has 93 threads.

The Xeon processor used in this experiment is E5-2687W with sandy bridge architecture. It has 16 physical cores, each supporting 2-way simultaneous multi-threading. Each core has 512KB L1 cache, 2MB L2 cache and 20MB L3 cache. Since there are more threads than logic cores, the workload balancing is done by OS. The 7110P Intel Xeon Phi coprocessor used in our experiments has 61 cores with 8GB RAM, each core runs at 1.1GHz and supports 4 hardware threads. Each core has 32 KB L1 cache and 512 KB L2 caches. Each cell processor has 6 Synergetic Processing Elements (SPE) and one PowerPC processor. Each SPE handles one thread. NVIDIA C2050 GPGPU is used in the experiment, which has 448 CUDA cores and 1288 GFLOP single precision peak performance

The execution time of one workload of BSB process is reported in Figure 11. The speedups with the respect to the sequential implementation on a single core processor are also reported in the same figure. As we can see, in average 7x speedups are achieved by parallel processing on multicore architectures. The reason that the performance improvement does not scale linearly with the level of parallelism is due to the contention for the memory bandwidth. More details about implementation and optimization of pattern matching layer can be found in [12].

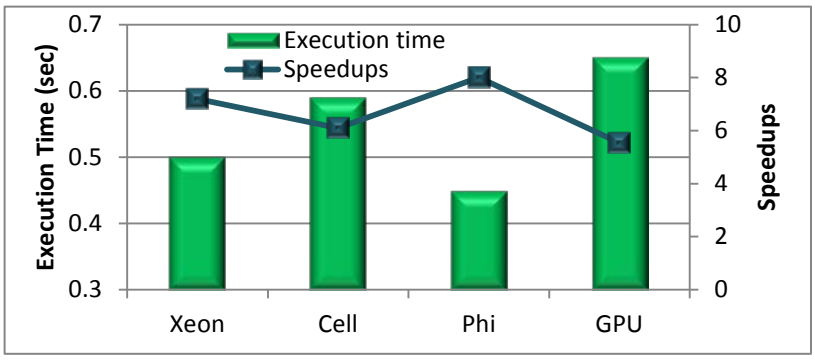


Figure 11 Comparison of BSB performance

4.1. Performance improvement in word confabulation layer

The original word confabulation layer stores dictionary in a hash table. All combinations of input letter candidates are checked against the dictionary and the valid word will be forwarded to the sentence confabulation layer. Using the trie data structure significantly reduced the search time, and improves the word confabulation performance. Table 3 compares the word confabulation time of old implementation to that of the new implementation when processing input images with different qualities. The clean image has the highest quality while the camera occluded image has the lowest quality. Lower quality input image leads to higher ambiguity in pattern matching. As the number of letter candidates increases, the complexity of the original implementation of word confabulation increases exponentially as it has to check all the combinations of the letter candidates. The new implementation has much lower complexity because it pruned many invalid combinations in advance. Furthermore, the hash table based dictionary storage in the original implementation has very poor memory locality, which degrades the performance even more.

Table 3 Improvement in word confabulation layer

	Word Confabulation Time (sec)		
	Clean image	Scan Occluded	Camera Occluded
Original implementation	310	2997	2483
New implementation	0.3	1.28	1.71

4.2. Performance improvement in sentence confabulation layer

The original sentence confabulation model maintains a separate knowledge link for each pair of source and target lexicons that has distance within +5 and -5. The new implementation merges all knowledge links between source and target lexicons that have the same relative position. We refer to the improved confabulation as circular model. The adoption of circular model not only reduces training and recall time, but also improves the accuracy of sentence completion. Table 4 shows that non-circular model takes about four times training effort more than the circular model, and 17.5% more recall time, but gives 13% lower recall accuracy.

Table 4 Comparison of non-circular and circular models

	Non-circular	Circular	Improvement (%)
Training time(s)	489180	144540	70.45%
Recall time(s)	6317.22	5207.83	17.56%
accuracy	54.95%	68.13%	13.18%

We also reduced the initialization time of sentence confabulation by loading the knowledge base in parallel. The size of sentence confabulation knowledge base is more than 7GB. Loading the knowledge base sequentially takes more than 83.9 seconds. A multi-thread implementation that loads the knowledge base in parallel can reduce the initialization time to 31 seconds and provides 2.7x speedups. More details about improvements in the sentence confabulation layer can be found in [11][13].

4.3. Accuracy improvement in sentence confabulation layer

Integrating the POS tag in confabulation model significantly improves the sentence confabulation accuracy. To evaluate the impact, the tag-assisted confabulation method is compared with no-tag confabulation at various noise levels. The noise level percentage means the ratio of characters in text with a 3-pixel wide horizontal strike. Note that with the size of original character being 15x15 pixels, a 3-pixel wide strike is almost equivalent to 20% distortion.

Figure 12 shows that no-tag sentence confabulation quickly collapse as noise level increases. This is because each test sentence contains on average 28 characters and we only consider the sentence correct if all of its characters are correct. The noise level at character level is compounded into character and word level ambiguity. Without semantic information, which provides an overall structure for each sentence, the success rate is expected to drop exponentially as noise level increase. Tag-assisted confabulation shows clear improvements over no-tag confabulation at all noise levels. The improvement is minor at low noise level, but significant at high noise level. Overall, tag-assisted confabulation improves success rate by 33% in average.

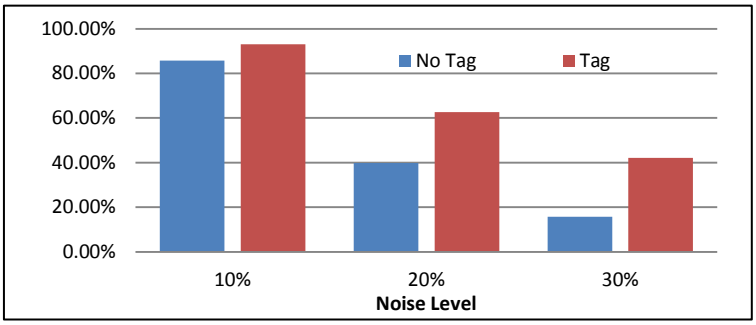


Figure 12 Accuracy comparison between tag-assisted and no-tag sentence confabulation

We also evaluated the impact of having segmentation, word pair and word triplet lexicons, assigning mutual information based weight to knowledge links, and enlarging the value of bandgap on the accuracy of Chinese sentence confabulation. Six different confabulations of sentence confabulation models are tested. Their configurations are listed in Table 5. Their recall accuracy is presented in Figure 13. The results show the best configuration of the confabulation model.

Table 5 Six configurations of Chinese sentence confabulation model

Configurations	Segmentation	Weight	Large Bandgap	Word pair	Word triplet
1					X
2	X				X
3	X	X			X
4	X		X		X
5	X	X	X		X
6	X	X	X	X	X

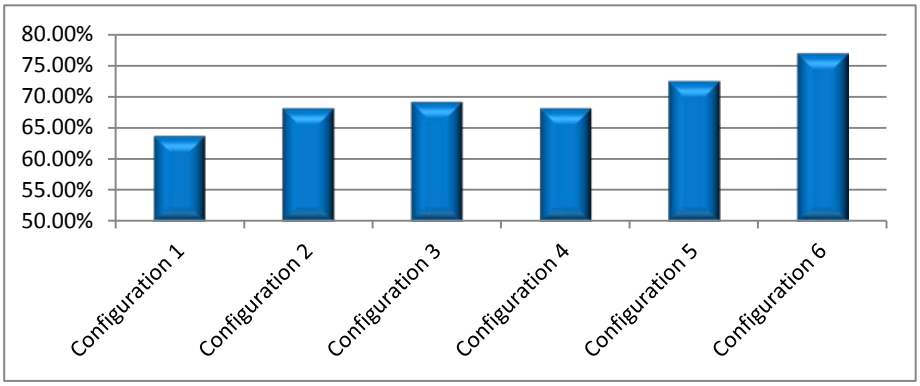


Figure 13 Sentence recall accuracy

4.4. Accuracy and performance of ITRS

In the last experiment, we compare the accuracy of ITRS with that of Tesseract. Developed initially at HP lab and now at Google, Tesseract is claimed to be the most accurate open source OCR engine available. Three sets of input images were tested. From high quality to low quality, they are: scanned clean images, scanned images with occlusions and camera images with occlusions. The word level and sentence level accuracy of both ITRS and Tesseract are given in Table 6. As we can see, with the reduced image quality, the accuracy of Tesseract degrades rapidly, while the performance of ITRS is more robust. Although Tesseract produces perfect recognition with given clean image, the ITRS is more reliable under noisy environment for low quality images.

Table 6 Compare word accuracy between ITRS and Tesseract

Input quality	Scanned clean images	Scanned images with occlusions	Camera images with occlusions
Tesseract	100%	93.1%	88.6%
ITRS (default)	97.6%	93.5%	90.9%
ITRS (best)	99.0%	94.8%	91.9%

Please note that, unlike Tesseract which recognize words and sentences solely based on image information, ITRS cannot guarantee the recognition of any word that is not in its dictionary. This is because the known words will always receive higher excitation than unknown words during sentence confabulation, which is analogy to human cognition process. If we exclude all proper nouns, such as the name of characters and locations, the word level accuracy of ITRS can be raised to 99%.

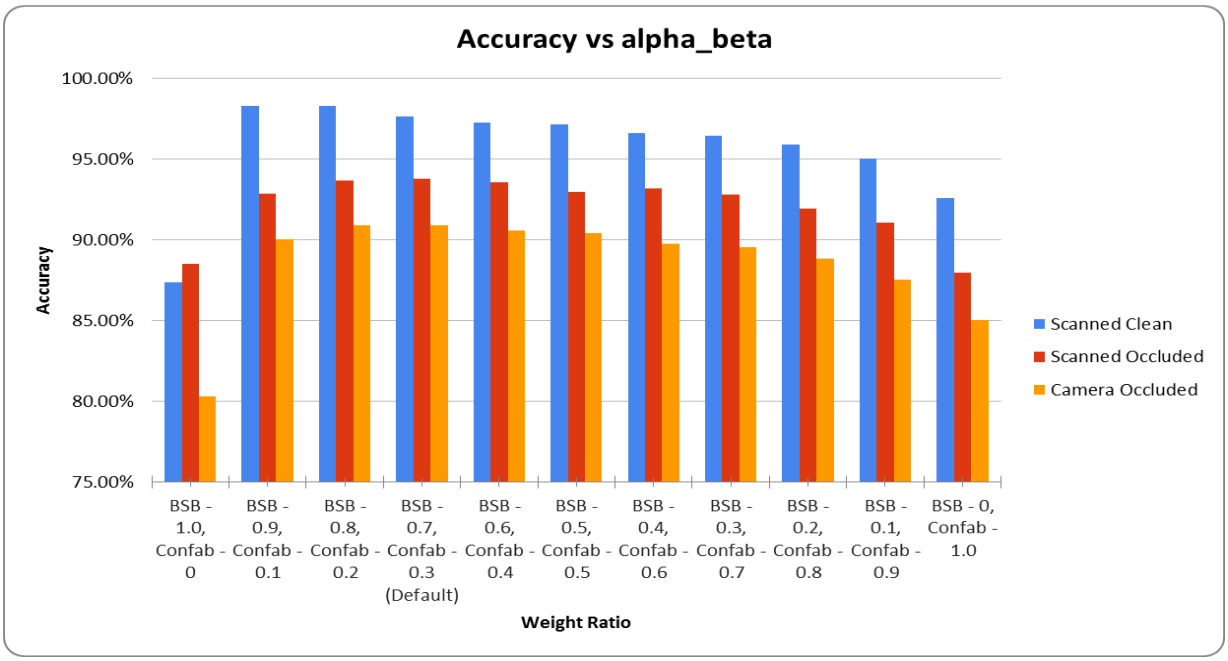


Figure 14 Adjusting the weight of image and language information affects the accuracy of ITRS

As shown in equation (3), the excitation level of a word in the sentence confabulation layer is a weighted sum of two components. One of them represents the likelihood of the word based on the input image; the other represents the likelihood of word based on the language context. The parameters α and β control the weight of image information and language knowledge. Adjusting the value of α and β affects the accuracy of ITRs. Figure 14 shows how the word accuracy changes as we vary the value of α and β from (1.0, 0) to (0, 1.0) at a step of 0.1. As we can see, completely ignore either the image inform or language information will lead to poor accuracy. Furthermore, for a clean image, we would like to rely more on the image information, and the best quality recognition happens when α and β are set at (0.9, 0.1); while for a low quality image, we would like to rely more on language information, and the best quality recognition happens when α and β are set at (0.7, 0.3).

Table 6 row 4 gives the word accuracy of ITRS under best setting of α and β parameters and with proper nouns excluded. As we can see that it has comparable recognition accuracy as Tesseract under clean images and better accuracy for images with poor quality. Finally, our experimental results show that the ITRS is able to recall words with occluded characters with more than 60% accuracy. Tesseract cannot recall any of these words.

5. CONCLUSIONS

In this project we developed an intelligent text recognition system that recognize document image based on word and sentence level knowledge. Its bottom layer performs pattern detection that matches the input image with stored pattern of characters; and its upper layer performs statistical inference that resolves the ambiguity in pattern matching. An open CV based image processing frontend is also developed that separates the text regions and extracts lines and characters.

In the future, we will further improve the accuracy of the sentence confabulation layer and pattern matching layer. In the sentence confabulation layer, we will investigate techniques that integrate semantic information with the language model. Current confabulation model selects word candidates only based on their grammatical relation. Semantic should be assigned in addition to POS tag to enhance those words that are semantically correlated. At BSB layer, we will investigate feature pooling techniques. Instead of consider the entire character image as one stored pattern, we will extract common features of the image and rely on feature pooling to detect character.

We will also extend the proposed brain inspired information processing frame work to other applications areas, such as speech recognition and image recognition. For those applications, a new set of lexicons will be defined and a new frontend should be developed for audio or image signal processing. The rest of the ITRS software can be reused, including training and recall procedures and the inter-processor communication framework.

6. REFERENCES

[1] R. Wray, C. Lebiere, P. Weinstein, K. Jha, J. Springer, T. Belding, B. Best, and V. Parunak, "Towards a Complete, Multi-level Cognitive Architecture," *Proc. of the International Conference for Cognitive Modeling*, 2007.

[2] R. S. Swenson, "Review of clinical and functional neuroscience," *Educational Review Manual in Neurology*, Castle Connolly Graduate Medical Publishing, 2006.

[3] J. A. Anderson, "An Introduction to Neural Networks," The MIT Press, 1995.

[4] 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* (ICME), June 2008.

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

[6] Qinru Qiu, D. Burns, M. Moore, R. Linderman, T. Renz and Q. Wu, "Accelerating Cogent Confabulation: an Exploration in the Architecture Design Space," *Proc. Of 2008 International Joint Conference on Neural Networks and IEEE World Congress on Computational Intelligence* (WCCI) , June 2008.

[7] Qinru Qiu, Q. Wu, M. Bishop, R. Pino, and R. W. Linderman, "A Parallel Neuromorphic Text Recognition System and Its Implementation on a Heterogeneous High Performance Computing Cluster," *IEEE Transactions on Computers*, Vol 62, No. 5, 2013.

[8] Toutanova, K.; Manning, C. D. "Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger," *Proc. Of the SIGDAT conference on Empirical methods in natural language processing and very large corpora*, 2000.

[9] Ratnaparkhi, A. "A Maximum Entropy Model for Part-of-Speech Tagging," *Proc. of the Empirical Methods in Natural Language Processing*, pp. 133-142, 1996.

[10] "Stanford Log-linear Part-Of-Speech Tagger," The Stanford Natural Language Processing Group, URL: http://nlp.stanford.edu/software/tagger.shtml, Accessed October, 2011.

[11] F. Yang, Qinru Qiu, M. Bishop, and Q. Wu, "Tag-assisted Sentence Confabulation for Intelligent Text Recognition," *Proc. Of Computational Intelligence for Security and Defense Applications* (CISDA), May, 2012.

[12] K. Ahmed, Qinru Qiu, P. Malani, M. Tamhankar, "Accelerating Pattern Matching in Neuromorphic Intelligent Text Recognition System Using Intel Xeon Phi Coprocessor," *Proc. International Joint Conference on Neural Networks* (IJCNN), 2014.

[13] 1. Z. Li, Qinru Qiu, M. Bishop and Q. Wu, "Completion and Parsing Chinese Sentences Using Cogent Confabulation," on *Proc. Of IEEE Symposium Series on Computational Intelligence* (SSCI), 2014.