Scalable Programming for the Analysis of Aphasia Transcripts

Paula Garcia and Vicki Hanson
Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14623 USA

***Abstract*—Technologies designed for individuals with non-fluent aphasia focus on digitizing speech therapy methods and generating speech. To improve these technologies, the language characteristics of individuals with non- fluent aphasia must be further understood. Language corpuses, such as the AphasiaBank, provide a promising solution for informing technology usability in terms of navigation, interface, and content decisions. As a tool for informing such work, this research investigates the viability of a flexible and scalable multi-threaded software program for the analysis of AphasiaBank transcripts. Results show that the program allows rapid analysis of all transcriptions by optimizing core functionality and minimizing the number of areas for synchronization. This research aims to improve the access to information, products, and services in technology for individuals with non-fluent aphasia.**

Keywords—Non-fluent Aphasia; Textual Analysis;

I. Introduction

Non-fluent aphasia is most commonly caused by stroke or head injuries, effecting language grammar and understanding of written text [1]. Individuals with non-fluent aphasia however, maintain phonological production and the comprehension of spoken language. The aim of this research is to facilitate automated language analysis of non-fluent aphasia to improve access to technologies and services. Currently used technologies and appliances that rely on textual comprehension and spoken fluency are largely inaccessible to individuals with non-fluent aphasia [2].

This study also supports the efforts of varying disciplines, including speech pathology, linguistics, and neuroscience, which have investigated how language attributes (such as phonology and syntax) impact spoken language fluency [1], [3], [4]. Automated language analysis serve as a tool for exploring how language attributes interact.

There are various text categorization software such as N-Gram, Wordle, and DocuBurst [5], [6]. These tools have been created to analyze texts and summarize or distinguish word metrics such as word frequency and synonyms. Although these tools provide insights towards the frequency and associations of words, they do not provide in depth information regarding the phonology and syntax of language in aphasia. In addition, none of these tools have been tailored to inconsistent texts with false starts, utterances, and incomplete phrases, all of which are common in non-fluent aphasia.

Figure 1: Sample Text Analysis and Visualization Tools

Spoken language corpora present promising opportunities for supporting accessibility in communication by providing comparable texts among a set of users/participants. This study tests the feasibility of a customized, scalable transcript analysis of individuals with aphasia, in order to provide actionable insights for creators of accessible technologies.

II. Methods

The developed program focuses on the parallelized analysis of interview transcripts from the crowdsourced language corpus, AphasiaBank [7]. AphasiaBank is a database of multimedia interviews among individuals with aphasia and language researchers [7]. This crowdsourced database includes transcripts from individuals throughout the United States, effectively providing language samples of people with non-fluent aphasia.

We began by testing the processing time of threads for files versus strings to ensure scalability of transcript processing at a growing rate of uploads. Scalability was also important in ensuring additional linguistic testing functions.

First, eighty-eight transcripts/files were tested for thread processing time on strings versus files. In the first scenario, the threads would be provided transcript lines/strings to process that had already been removed of utterances and false starts. In the latter scenario, threads would be provided full transcript files, which would each, have to be opened, read, and processed. When sending over files, the average processing time was 0.25 seconds. When sending over strings for processing, the average completion time was 1.50 seconds. For this reason, the program layout was created as follows, focusing on file transferring:

- S_1: File manager locates all available files and stores them in an array.

- S_2: Initiates threads based on the number of files and processors available. Each thread is provided a copy of all the files with a range to read.
- S_3: Thread reads the files and tries to find the number of syllables for each word (S_4). When all words have been processed, it updates all participants' information (S_5).
- S_4: Calculates the number of syllables and creates an object of S_5.
- S_5: Stores various attributes for a word (i.e syllable count, location, length of sentence, etc.).

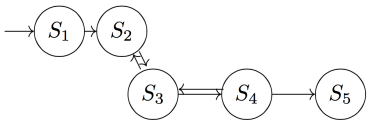

Figure 2. Graph of Program Classes and Interactions

Next, we tested the efficacy of the program to confirm scalability and time-efficiency. We generated a bootstrapped dataset to simulate larger file volumes beyond the current 88 transcripts available in the database. Duplicate files were randomly generated from the 88 files. The testing environment is outlined in Table I.

TABLE I. TESTING ENVIRONMENT SPECIFICATIONS

	Details
Model	iMac, late 2012
Processor	2.9 GHz Intel Core i5
Memory	32 GB 1333 MHz DDR3
OS	OSX Yosemite v. 10.10.3

A total of 1,750 files were used in the data set, with varying intervals. The specific file sizes and strings/lines are detailed in Table II. These increments were created to calculate the associated speedup and efficiency values for parallel execution.

TABLE II. SIMULATED FILE SET FOR PROGRAM TESTING

File Count	**Total Size**	**Number of Lines**
500	4.5 MB	85,217
750	6.4 MB	123,507
1,000	7.9 MB	158,479
1,500	12.7 MB	237,472
1,750	16.2 MB	294,948

III. RESULTS AND DISCUSSION

The program was run on various data sizes and threads, as shown in Table III. For 500 files, the speed increased as the number of threads increased from 0.04 seconds for one thread, to 0.37 seconds for four threads. In this case, the file load does not overcome the communication delays in parallel execution. A decrease in file processing duration, however, was observed as early as 750 files where each thread was able to parse a file (reading and removing single-letter words and symbols) in under a second.

TABLE III. FILE PROCESSING SPEED (SECONDS) BY NUMBER OF THREADS

N=500	(1, 0.04)	(2, 0.33)	(3, 0.34)	(4, 0.37)
N=750	(1, 0.93)	(2, 0.72)	(3, 0.62)	(4, 0.63)
N=1,000	(1, 1.42)	(2, 1.02)	(3, 0.88)	(4, 0.81)
N=1,500	(1, 2.25)	(2, 1.49)	(3, 1.40)	(4, 1.31)
N=1,750	(1, 3.16)	(2, 2.26)	(3, 1.88)	(4, 1.53)

The total processing time for syllable calculation, phrase length, and word location is summarized in Fig. 3. Generally, more files can be processed in a shorter period of time as the number of threads increases. The only exceptions are the completion times for three and four threads on 1,750 files. For three and four threads, the completion times are 19.14 and 19.54 seconds where two threads is 18.49 seconds. This does not follow the trend from all previous completion times.

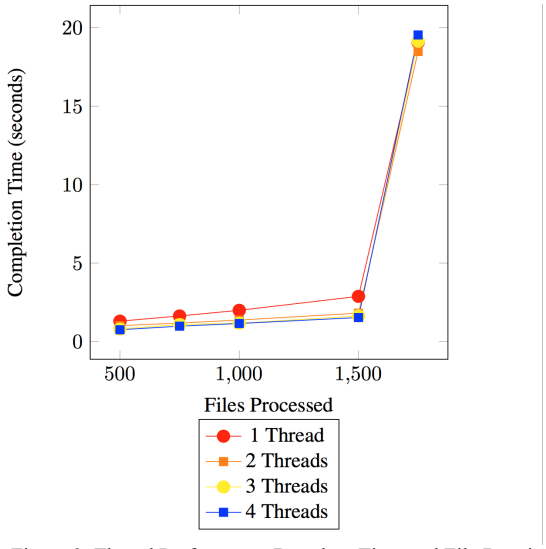

Figure 3. Thread Performance Based on Time and File Density

To further investigate this occurrence, we evaluated the file parse times and word/line distribution by thread. As seen in Table III, for 1,750 files the file parse times are gradually decreasing which is inconsistent with the overall file processing speeds in Fig. 3. The line and word count for 1,750 files per number of threads were typically equally divided, as seen in Table IV. The threads had at maximum, a +-10 word difference. Taking these results into account, the next area to investigate is the influence of memory on processing time.

TABLE IV. DIVISION OF LINE AND WORD BY PROCESSOR

Number of processors	**Line Count**	**Word Count**
2	139,655 155,293	4,380 4,370
3	88,777 99,859 106,312	2,920 2,920 29,10
4	64,746 73,460 74,909 81,833	2,190 2,190 2,190 2,180

As shown, a delay in processing may be attributed to the fact that 1,750 files were approaching the memory processing limits of the machine. In other words, the processing times may

become less effective as the computer reaches its peak memory allocation. The speedup (time reduction through parallelism) and efficiency (processor optimization/utilization) calculations are specified in Table V. Speedup and efficiency are inversely related: where speedup increases, efficiency decreases [8]. Table V indicates the speedup and efficiency for four processors during varying file volumes. Speedup was calculated by dividing the processing time for one processor by the processing time for the highest number of processors tested (P_{max}=4). The corresponding efficiency value per processor was calculated by dividing the speedup values by the highest number of processors tested.

TABLE V. SPEEDUP AND EFFICIENCY FOR FOUR PROCESSORS BY FILE VOLUME

N		Speedup		Efficiency
500	$\frac{1.30}{0.75}$	**1.73**	$\frac{1.73}{4}$	**0.43**
750	$\frac{1.63}{0.98}$	**1.66**	$\frac{1.66}{4}$	**0.415**
1,000	$\frac{1.99}{1.15}$	**1.73**	$\frac{1.73}{4}$	**0.43**
1,500	$\frac{2.88}{1.53}$	**1.88**	$\frac{1.88}{4}$	**0.47**
1,750	$\frac{19.07}{19.54}$	**0.975**	$\frac{0.975}{4}$	**0.24**

Efficiency rates may be optimized in the future by adding additional linguistic testing beyond syllable segmentation, word count and location. Efficiency scales may also be improved by the addition of visualization of the text analysis. The addition of visualizations would allow individuals with non-fluent aphasia to obtain language feedback without excess amounts of text.

A. Limitations

The number of files was divided in increments up to 1,750 because no more than 1,800 files could be processed with the computer's given memory. This may have impacted the efficiency and speedup rates for 1,500 and 1,750 files. The testing environment can be seen in Table I. Future studies will be generated on distributed systems, which may adequately handle the number of files as the database grows, displaying higher efficiency values.

The findings for speedup were calculated by the execution time of a parallel program of one thread. A more ideal calculation would be based on the sequential time, to remove any delays associated with parallel programming. However, since most of the processing is done though multi-threading there are minimal areas where functions are completed synchronously and sequentially. Sequential functions are only done when accessing files and dividing the files by the available threads (execution time=0.014 seconds). This suggests that one thread execution time in comparison to a sequential execution time, may be similar in value.

Lastly, the current findings are based on a simulated, bootstrapped sample. More precise findings will be achieved as the database increases.

IV. CONCLUSIONS AND FUTURE WORK

Overall, the program shows the potential to be scalable as the number of files increases. The speedup rates gradually increase, until approaching memory capacity. The response lengths among interviews are similar in size and word length, supporting load balancing. However, more exploration is needed to verify that the changes in completion time are truly attributed to the limits of the testing environment.

Furthermore, the completion times are low, for 1,750 files four processors' execution time was 19.54 seconds. The results for speedup (ideally: ~number of processors, also referred to as linear speedup) and efficiency (ideally: ~1) are not as powerful as they may be but as more linguistic testing functions and visualizations are included into the program, the scale of these results may improve. Fast completion times for processing permit the inclusion of additional linguistic analyses and visualizations.

Our future efforts are focused on adding additional linguistic attributes as well as a visual representation of the data.

ACKNOWLEDGMENT

We would like to thank Minseok Kwon, at Rochester Institute of Technology, for guidance in this research.

REFERENCES

[1] J. Galliers, S. Wilson, A. Roper, N. Cocks, J. Marshall, S. Muscroft, and T. Pring, "Words are not enough: empowering people with aphasia in the design process," *Proc. 12th Particip. Des. Conf. Res. Pap. - Vol. 1*, pp. 51–60, 2012.

[2] A. Roper, "Accessibility of Computer Therapy and Technology for People with Aphasia," *ACM SIGACCESS Access. Comput.*, no. 108, pp. 50–53, 2014.

[3] N. Braber, K. Patterson, K. Ellis, and M. a. Lambon Ralph, "The relationship between phonological and morphological deficits in Broca's aphasia: Further evidence from errors in verb inflection," *Brain Lang.*, vol. 92, no. 3, pp. 278–287, 2005.

[4] R. Chapey and B. Hallowell, "Language intervention strategies in adult aphasia," in *Language intervention strategies in adult aphasia*, 2001, pp. 472–486.

[5] C. Collins, S. Carpendale, and G. Penn, "Docuburst: Visualizing document content using language structure," *Comput. Graph. forum*, vol. 28, no. 3, pp. 1039–1046, 2009.

[6] F. B. Viégas, M. Wattenberg, and J. Feinberg, "Participatory visualization with wordle," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 6, pp. 1137–1144, 2009.

[7] B. MacWhinney, D. Fromm, M. Forbes, and A. Holland, "AphasiaBank: Methods for studying discourse," *Aphasiology*, vol. 25, no. 11, pp. 1286–1307, 2011.

[8] D. L. Eager, J. Zahorjan, and E. D. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 408–423, 1989.