EASYREAD

Bryan Kane

The George Washington University
Department of Computer Science

Senior Design 2014

THE GEORGE WASHINGTON UNIVERSITY
WASHINGTON, DC

Project Overview

EasyRead, a web-based tool, assists educators with the simplification of news articles and other documents. Designed for the adult education community, EasyRead allows teachers to find material that is more relevant for their older students, and then modify the text with EasyRead's help to reduce the difficulty of the passage.

The illiteracy rate amongst adults in the United States is 14%, with another 7% reading below a fifth-grade reading level. Yet the resources designed to help adults learn to read are hard-to-find, outdated, and uninteresting. In order to learn how to read, students must have the motivation to practice reading often. However, if students are given resources that are designed for younger children, or are simply uninteresting, this necessary motivation simply won't exist.

EasyRead's attempt to solve this problem involves taking interesting materials, such as current newspaper articles about any given topic, and providing assistance with the article to lower the reading level and help students get past trouble areas. By reducing the barrier for instructors to create up-to-date materials, students will benefit from more interesting materials that they'll be more likely to read, and thus will improve simply due to the fact that they're more engaged in their learning.

The EasyRead system consists of various "actions" that the user can perform on the document they've chosen, ranging from the addition of contextual information, such as definitions and biographies of people mentioned in the document, automatic replacement of difficult words with easier synonyms, and the ability to remove difficult passages completely. These actions are very modular, and are designed to guide users through a step-by-step process, eventually resulting in a greatly simplified document. While as many steps are designed to be as automated as possible, some user interaction is still required in making certain decisions as to the difficulty of certain passages and the best actions to take for simplification. However, all user interaction is guided through the EasyRead system, and users are never left completely on their own in determining which actions to take.

The typical end-result of an EasyRead processed document contains a dozen replacements of difficult words with easier alternatives, five or six detected entities (people, locations, or organizations) with information added to the margins, and a handful of words that are defined in the margins. Once a user has completed their work in the EasyRead system, they can export their finalized document to a print-view, with the contextual information that they added in the margins.

Technical Design

The EasyRead system takes a service-oriented architecture approach to its design, with multiple systems, each serving its own task. This architecture keeps the system very modular, and allows for expansion of the system without polluting the code base, while simultaneously allowing each task to be written in the best language and system for the job.

Text-Analysis Engine

The main component of the EasyRead system is the text-analysis engine. This system, written in the Scala language and utilizing the Play! Framework, is designed to be a very responsive, high-throughput system capable of performing hundreds of analysis tasks on a document simultaneously. The fast processing speed provided by Scala's asynchronous engine running on the Java Virtual Machine (JVM) is critical for returning results to the user in a matter of seconds, rather than the minutes that some other systems would take.

The text-analysis system also makes heavy use of the Stanford NLP Parser. When compared to the Natural-Language Tool Kit (NLTK) system available in Python, the Stanford Parser is both faster and returns more information than the NLTK. However, this comes at a cost of a very high memory footprint (requiring upwards of 2 gigabytes), and a multiple-minutes startup time before the server can accept any incoming documents. Given the usage of the EasyRead system, prioritizing the fast responses for users is a priority, thus leading to the Stanford NLP Parser.

In order to further reduce wait-times for users after the document analysis was initialized, the system makes use of WebSockets for concurrent communication between the client and the server. Text analysis tasks on the server are broken up into different actions, each with its own independent set of inputs and outputs. As each of these tasks finishes, it is able to send its results directly to the user's browser, thus allowing users to see the text-analysis progress as it is completed, and resulting in a seemingly faster experience. This also reduces strain on the server, because it is able to finish up a task without having to wait for all other tasks to run, and is then available to take on further requests immediately.

Analyzers in the Text-Analysis Engine

Each individual analyzer on the server is capable of running completely independently, reducing any dependencies that could cause a backlog on the server. A few of these analyzers are detailed below:

Word Difficulty Analyzer:

This analyzer makes use of two sources of word difficulty: age-of-acquisition lists, and word-frequency lists. The age-of-acquisition list consists of a simple key-value list between a word, and the expected age that a person would know that word. For example, the word "read" has an expected age-of-acquisition of 4.11, while the word "cardiology," a much more difficult word, has an expected age of 13.58. By looking up the age-of-acquisition of each word in a document, words that are higher than the chosen reading level can be pointed out to the user, allowing the replacement of the difficult word with an easier one.

In addition to the age-of-acquisition list, EasyRead also uses a frequency list to measure word difficulty. This list, much like the age-of-acquisition list, contains a key-value pair between a word and a frequency score. This score was calculated by counting the number of times a word occurs in the Corpus of Contemporary American English, a large corpus of over 100,000 documents, such as books, newspaper articles, movie scripts, and research papers, published in the United States over the past 30 years. This measure makes the [generally-accepted] assumption that words that show up more across written text are more likely to be known by earlier readers, and rarer words are likely more difficult.

Through a matching of these two lists, EasyRead is able to calculate which words would cause a problem for early readers, and allow the user to take an appropriate action.

Synonym Replacement Analyzer:

This analyzer makes use of the same lists as the word difficulty analyzer, except it goes one step further and attempts to replace difficult words with simpler synonyms automatically. Once a list of difficult words is discovered and stop-words and entities are removed, EasyRead sends each word to a thesaurus, and gathers a list of possible synonym replacements. Once that is complete, EasyRead then makes use of the Microsoft Bing Web N-Gram Service API to determine the best match. This API, given a list of five-word-phrases, returns a score of how often the phrase appears across the Internet. Easy synonym (possible replacement word) is replaced with the word in a phrase with the two prior words and the two words succeeding it, to get a list of possible phrase replacements, and determining which one is the best match. The Bing API is extremely useful in determining which synonyms would be a good match, and which don't fit into a sentence at all. Given the fact that it is sometimes difficult to determine a word's part-of-speech in a sentence, it is very important to ensure that, for example, a noun is not replaced with a verb in a sentence.

Entity Detection Analyzer:

This analyzer has one simple task: to detect entities inside of a document. The entities that it can detect include people's names, locations, and the names of organizations, generally determined by the sentence structure and capitalization scheme. The Stanford NLP Parser's entity-detection engine is responsible for parsing the article, and then wrapping all detected entities in XML tags. The EasyRead analyzer can then parse the XML document to extract these entities from the XML tags, and return them, grouped by category.

Sentence Relevancy Analyzer:

The sentence relevancy analyzer makes use of a third-party library called TextTeaser, which is used to determine the five most relevant sentences in a newspaper article. However, instead of using the library in the traditional way to determine the most relevant sentences, EasyRead reverses the search, to determine the *least* relevant sentences in an article. This is calculated by comparing the words in a sentence with the words in the rest of an article and the title.

Front-End User Interface

While most of the heavy analysis is conducted on the backend server, the front-end user interface is responsible for displaying the content to the user in a clear and efficient way. All of the interactions between the backend and frontend are conducted via WebSockets to provide greater responsiveness and reduce the time to the first result. Once a user has selected an article on the frontend, the contents of the article are sent to the backend to be analyzed, and as each analyzer completes, the responses are loaded onto the page. As a user navigates through each available "action" (which is a result of each analyzer), the user can choose different actions to take to simplify the text.

For example, on the word-difficulty action, users can view the most difficult words highlighted in either red or yellow (depending on difficulty). When a user clicks on one of the difficult words, he or she will be presented with a list of possible definitions that could apply to the word, as well as synonyms that could be used to replace the word. If a user clicks on the “add definition” button, the definition of the word will be added to the document’s margins upon export. If a synonym is chosen instead, the word will be directly replaced inside the document.

On the entity detection page, users will be presented with a similar interaction on selecting an entity. However, since definitions are not available for people or places, the Google FreeBase API is used to retrieve a brief description of a chosen entity, as well as “quick facts” that could provide additional context to entity in the document.

Project timeline / challenges

The actual implementation of the project was not very difficult, and only involved putting in enough time to implement the system. However, given the nature of the senior-design class, there were many distractions throughout the implementation time that took effort away from building a quality product, and instead focusing on how to present a sub-par project.

Knowing how to develop the product took experience that was gained through previous internships and independent projects, and for the majority of the implementation period, only involved appropriating already-gained skills to apply to the specific problem at hand.

The first month or two of implementation began with a very ambitious plan, involving advanced machine-learning and natural-language-processing. However, as the project timeline continued, it became apparent that these skills would be impossible to gain independently, given the limited time available for the project.

Additionally, many roadblocks that came up during the project that ended up being small syntax errors or slight lapses in planning turned into major setbacks. Had the project been a group project, many of these issues could have been solved with a simple discussion or second set of eyes, thus allowing much greater progress in a shorter period.