A Framework for an Interactive Word-Cloud Approach for Visual Analysis of Digital Text using the Natural Language Toolkit

Musa Jafar
mjafar@mail.wtamu.edu

Jeffry Babb
jbabb@mail.wtamu.edu

Kareem Dana
kdana@mail.wtamu.edu

Department of Computer Information and Decision Management
West Texas A&M University
Canyon, TX 79016

Abstract

In this paper we present a web-based framework which supports the visual analysis of digital text. Such analysis is of increasing importance as growth of data in our contemporary computing environment has outstretched conventional means of presentation and analysis. Hence, we present a framework that assists in the visual presentation and digest of textual corpora. The framework presents a front-end as a set of web-pages which employ a word-cloud approach to present and manipulate the underlying digital text for the purpose of understanding and analyzing its content. The framework's back-end is an Apache-Based Django framework that uses the Python programming language to access the Natural Language Processing Toolkit's (NLTK 2.0) capabilities for the parsing and analysis of digital text. We also present examples of our framework's practical application. Ideally, our work will allow a humanities researcher, or similar non-technical professional, to analyze and manipulate text without needing to know the particulars of NLTK or Python, and without having to read the whole digital text.

Keywords: Natural Language Processing, NLTK, Word cloud, data visualization, corpus linguistics

1. INTRODUCTION

A poignant mark of our progress towards civilization has certainly been the ability to develop language and record this language in written means. In this sense, written language becomes a tangible artifact that preserves the thoughts, habits, and norms of our civilizing. As a form of communication, the written word is a foundational basis of the transactions that allow for societies to function. In the modern era of computing, machines have greatly assisted in digitization of this written language and the proliferation of writing. Indeed, there is such a great volume of digitized text that sub-disciplines such as data mining have emerged in hopes that sense can be made, on the whole, of a staggering volume of text (Jacobs, 2009).

The "plainest of text" becomes beautiful when analyzed because it says so much about the language and the world that speakers inhibit (Norvig, 2009). In this sense, we can imagine that our bodies of text are rather deep repositories: William Shakespeare's work is not just a body of literature, a New York Times article is not just an opinion, a Supreme Court of the United States judgment is not just an article in constitutional law, a customer's review of a product is not merely that individual's sentiment. When digitized, each of these texts can be thought of as data available for exploration, analysis, and to be reasoned about; therein lies the beauty of these texts. Towards this end, most of the text that has been published in the past 10 years is available in digital format. Moreover, ancient and historical texts are also being digitized. For example, Google's initiative to digitize over 15-million books created a wealth of data in digital format that is ready for interactive analysis (Jean-Babtiste, 2011).

Our objective is to provide a tool where a "non-technical" researcher or practitioner in the humanities (such as a journalist, etc.), or in any non-technical discipline, may analyze a body of text without deep knowledge of the underlying technologies used to implement the framework and without having to read the text itself. This last aim is perhaps the most compelling: what can visualization of text tell without having to actually examine the text?

2. RELATED WORK

In recent years, "word clouds" (sometimes referred to as tag clouds) were popularized in work of Viegas and Wattenberg in their the *Many Eyes* and *Wordle* projects (Viegas et al., 2007, 2009). These projects utilize manipulations of font-size, word placement, and word color to display the frequency distribution of words within a document. Another characteristic of these projects is the minimal interactivity afforded, such as the exclusion of words, modification of layout, and direction and color management (Figure 1).

Viegas et al. (2009: 1137) state: "…users seem to revel in the possible applications of word clouds, scientists wordle genetic functions, fans wordle music videos, … spouses wordle love letters, … wordles have made their way into corporate PowerPoint slides and houses of worship; they have adorned T-shirts, magazine covers, … have graced posters, scrapbooks, birthday cards, and valentines."

This work bears testimony to the wide-spread appeal of word clouds and the degree to which people have used them in many aspects of life. Sharma et al. (2012) used a word-cloud approach to infer the social connections and networks (a who-is-who approach) to highlight the characteristics of admired Twitter users (build a profile) by exploiting the metadata in the lists feature of a given admired user. The key words in the cloud provide hyperlinks to more searches on Twitter. Figure 2 shows the output of the Who-is-Who on Twitter when used to obtain more information about "Sir" Tim-Berners-Lee.

Baroukh et al. (2011) used a "WordCram" (Oesper, 2011) word-cloud to quickly summarize knowledge of biological terms. Figure 3 shows the output when we used WordCram to generate a word-cloud of the Supreme Court of the United State's decision on the Affordable Care Act. As WordCram is Processing-based (Processing being a Java library and framework for visualization), it is not a natural fit for a solution that should be browser-based and which relies on the AJAX capabilities of modern web browsers.

Kim et al. (2011) used a novel and network-oriented "node-link" approach to analyze test using word-clouds where the nodes represent entities or keywords and the links (or edges) represented the relationship or co-occurrences between the entities to build word-clouds of the nodes and their relationships.

For now, it is clear that the aesthetics of presentation and the management of layout and placement of the other word-cloud solutions are more appealing than ours. However, as our work progresses, the aesthetic and layout issues will be addressed. The advantage of our approach is in the richer interactivity possibilities for users. Our solution does not merely supply stand-alone web-page interface that heavily utilizes JavaScript or a standalone Java applet. Rather, our solution approach heavily relies on back-end processing using the Natural Language Toolkit (NLTK - http://nltk.org/) to analyze the text beyond its presentation as a word-cloud. It provides a user interface to the NLTK package for the purpose of analyzing the text and inferring knowledge about it. Figure 4 shows the output when we used our framework to generate

a word-cloud of the supreme-court decision on the Affordable Care Act.

3. OUR SOLUTION APPROACH

As stated previously, the NLTK constitutes the centerpiece of our solution approach and framework. The advantages that the NLTK provides for our solution approach is that it harnesses the ease-of-use and power of the Python programming language as well as a very capable library for accessing various corpora and lexical resources for the classification, tokenization, stemming, tagging, parsing, and semantic reasoning required as a prerequisite for the visual analysis of digitized text. Thus, we require a component like the NLTK as a precursor for successful visualization as our visualization engine requires the outputs of computational linguistics.

Figure 5 shows the control center of our framework after loading a text document. With our framework, we can browse and upload new files for analysis and add that new file to the list of available digital documents. We can select a document to analyze, by selecting "Generate Word Cloud," in which we opt to utilize a one-gram (default), two-gram, or a three-gram cloud. In computational linguistics, an n-gram is the contiguous sequence of phonemes, syllables, letters, or words that are considered to "go together" sequentially (for instance, "I love you" is a three-gram sequence). For this transformation to occur, all documents are pre-processed in our framework using NLTK. Once tokenized into grams, we can use our visual interface to highlight a token (in this case the three-gram "You need a" to see all the sentences of the token. We can also perform a regular expression search to find all of its usage in the body of text and the corresponding word-cloud tags.

What sets our solution approach apart from others mentioned here is that it uses the word-cloud as an interactive data analysis tool. Thus, our Django-based interface (both the administrative and user interfaces) serve as the front-end to an NLTK-driven back-end running on a server. This works such that, after a document is rendered, we use AJAX to communicate with the NLTK backend to facilitate a user's ad-hoc queries. In addition to supporting, one-, two-, and three-gram word analysis (tokenization), we further allow a user to drill-down to the underlying sentences of the tokens and to selectively filter out the x-gram that is being displayed via mouse-driven interaction with the visualization graphics directly.

4. OUR SOLUTION ARCHITECTURE

Figure 6 illustrates the architecture of the NLTK Command Center. One of our design goals was to make the NLTK more visual and more accessible to the end user. To that end, we designed the NLTK Command Center as a web application. The frontend runs in a browser that utilizes JavaScript, AJAX, jQuery, HTML5 (specifically HTML5 Canvas element), and CSS. These latest web technologies enabled us to write a fully-featured, graphical, and interactive web application. We implemented an existing HTML 5/Canvas word cloud library named "HTML 5 Word Cloud" written by Chien (2012).

The backend architecture was dictated by the fact that the NLTK is a Python library. Accordingly, we used Django, which is a Python-based web framework, as it can directly interface with the NLTK, which is also Python-based. For data persistence, the MySQL database server stores user session information and file metadata. We utilize the Apache web server to host our Django application on an Ubuntu Linux server. The Apache module mod_wsgi acts as glue between Apache and Python in the same manner as mod_php or mod_cgi; this module would be needed for any Python web application. Despite our entirely open-source implementation approach, there is nothing to prevent the deployment of our solution on any other platform that Django supports, such as Windows and Internet Information Server (IIS).

One of the major design challenges we faced involved the performance and scalability of the NLTK library. During our testing, we should note that processing a large body of text to be very CPU intensive. In fact, the initial loading of an NLTK corpus can take several seconds or longer depending on the corpus being used. By default, mod_wsgi loads a new instance of the Python interpreter executable for each Apache worker process. This translates to each visitor of the web site incurring the same delay. We worked around this problem by configuring mod_wsgi to instantiate just one Python process and share it amongst every request. This prevented the initial loading delay from occurring for subsequent client requests. However, this

implementation may lead to scalability issues down the road if the user base grows. However, there are other scalability solutions including Platform-as-a-Service (Paas) and Infrastructure-as-a-Service (IaaS) options. Accordingly, we believe that redesigning the architecture to support the Google App Engine or another cloud based service could be useful for the future.

5. IMPLEMENTATION DETAILS

When a user first connects to the NLTK Command Center website they are presented with user interface that allow them to perform various NLTK tasks through a GUI (Figure 5). This is facilitated by JavaScript running on the client's web browser that sends an AJAX call to the web server, and ultimately to the NLTK, which then performs the task and sends the results back serialized as JSON. We then use JavaScript to display the results graphically. Currently we provide and expose the following python methods which provide AJAX-oriented NLTK services in our framework:

- **`upload_file()`** – This is an HTTP POST command that uploads a file from the user's computer to the server. After uploading said file, the user can select it from a list of available files and perform various text analyses using the NLTK.

- **`word_frequency(file_id, ngram)`** – This function analyzes a text and returns the frequency of words as a list. It can compute the frequency of one-, two-, or three-gram phrases as specified by the ngram parameter. Two- and three-gram items will return the frequency of two- word and three-word phrases respectively. This AJAX call is used in to facilitate the drawing of the word cloud.

- **`get_sentences(file_id, phrase)`** – This AJAX-driven function allows the user to right-click on a word or phrase in the word cloud and request a list of all the sentences that word or phrase appears in. The NLTK processes each sentence of the text defined by file_id and looks for phrase. The list of sentences is then returned as JSON and displayed in the browser. The phrase is highlighted in each sentence while it is displayed in the word cloud visualization.

- **`regex_search(file_id, pattern)`** – The user can search the text using regular expressions. This AJAX call asks the NLTK to perform the regular expression and return all matches within the text. The matches are displayed in a drop down box and the user can select them. When selected, they will be highlighted in the word cloud.

- **`execute_nltk_command(command)`** - The user can enter any valid NLTK/Python command and the server will execute it and simply dump the standard output to the browser. This capability is mostly used for debugging but is available and can be useful for advanced NLTK analysis that is not yet implemented in the GUI.

There is additional functionality and logic programmed directly into JavaScript. Whenever possible, JavaScript is used to execute tasks directly on the web browser without communicating to the server through AJAX. This improves responsiveness and scalability. The biggest example of this is drawing the word cloud itself. We implemented an existing HTML 5/Canvas word cloud library named "HTML 5 Word Cloud" written by Chien (2012). As the library is freely available under the open-source MIT License, we appropriated just the HTML 5/Canvas JavaScript drawing routines and made two important changes. First, we added a two-dimensional `wordMap[][]` array that maps x and y pixel coordinates to the location of a specific word on the canvas. This array is filled in once while the word cloud is being drawn. This enabled us to add interactive functionality to the word cloud – that is, a user can right-click with their pointing device on the word cloud and it will know which X-gram word was selected. A second array, `pixelMap[]`, is an associative array that maps words to pixel coordinates. This enables us to allow a user to search for a word within the word cloud. With this functionality, we implemented the following features solely in JavaScript on the client-side:

- **`highlight_word(word)` – Coupled with the `regex_search(file_id, pattern)` above, the user can select a word, phrase, or several words and using the `pixelMap[]` array JavaScript will highlight those words in the word cloud.**

- **`exclude_word(word)` – The user can click on a word in the word cloud and**

exclude it. The wordMap[] array is used to determine which word was clicked. The word cloud is then redrawn with the specified word removed from the list. This is useful if there is one non-important word that appears so often it diminishes the weight of other words.

- **draw_wordcloud(list) – Given a list of words and frequencies, this draws the word cloud on the HTML 5 Canvas.**

6. SUMMARY AND FUTURE WORK

With our word cloud framework, our objective is to create tools that allow users to be engaged in the discovery of hidden characteristics and meanings ensconced within digital text. Facilitating knowledge through the design and implementation of information systems has long been a focus for the information systems discipline and, as such, we see the need for improving information interfaces with information visualization. Toward this end, we have developed a prototype implementation that allows those with minimum computing knowledge to be able to analyze digital text as data by seeing that data and by manipulating what they see. This this sense, a data visualization tool such as our prototype system constitutes a form of decision support system.

For instance, we envision the humanities researcher who may be well equipped to understand intrinsic and underlying literary and historical context of the text, but is inhibited from analyzing the data of the text by its sheer volume. Furthermore, such users may also find that their lack of expertise in utilizing the tools that can reduce this data leaves such analytic opportunities out of reach. We anticipate that the outputs of our research will allow for growth and impact not only in our field, but also in the humanities, in business, and in any areas where digitized text is available and text and data analysis is important. In summary, our framework should allow a user to analyze text without reading it. While this may seem counter-intuitive, decisions on how to use precious time in extracting value from text may be improved by "seeing" a text before reading it.

7. REFERENCES

Baroukh, C., Jenkins, S. L., Dannenfelser, R. and Ma'ayan A. (2011). Genes2WordCloud: a quick way to identify biological themes from gene lists and free text. *Source Code for Biology and Medicine*, 6(15).

Chien, T, G. (2012). HTML 5 Word Cloud. http://timc.idv.tw/wordcloud/en/ last accessed Aug-11-2012.

Jacobs, A. (2009) The Pathologies of Big Data, ACM Queue, 7(6), 10.

Jean-Baptiste, M., et.al. (2011). Quantitative Analysis of Culture Using Millions of Digitized Books. *Science*, 332, 176.

Kim K., Ko, S. Elmqvist, N., and Ebert, D. (2011) WordBridge: Using Composite Tag Clouds to Node-Link Diagrams for Visualizing Content and Relations in Text Corpora. *44th Hawaii International Conference on System Sciences (HICSS)*.

Norvig, Peter(2009). Natural Language Corpus Data, in Segaran, T. and Hammerbacher, J (eds.) *Beautiful Data* (219-240), O'Reilly Pub.

Oesper, L., Meico, D., Isserlin, R. and Bader, G. D. (2011) WordCloud: Cytoscape plugin to create a visual semantic summary of networks. *Source Code for Biology and Medicine*, 6(7).

Sharma, N., Ghosh, S., Benevnuto, F., Ganguly, N. and Gummadi, K. P. (2012). Inferring Who-is-Who in the Twitter Social Network. *ACM SIGCOMM Workshop on Online Social Networks (WOSN'12)*, Helisinki, Finland.

Viegas, F. B., Wattenberg, M., Ham F. V. Kriss J. and McKeon M. (2007). Many Eyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics*, 13(6).

Viegas, F. B., Wattenberg, M., Feinberg J. (2009). Participatory Visualization with Wordle. *IEEE Transactions on Visualization and Computer Graphics*, 15(6).

Yatani, K., Novati, M., Trusty, A., and Troung, K.N. (2011) Analysis of Adjective-Noun Word Pair Extraction Methods for Online Review Summarization. *Proceedings of the 22nd International Join Conference on Artificial Intelligence*.

ISSN: 2167-1508
v5 n2240

Figures

Figure 1 A Wordle word-cloud visualization of the supreme-court decision on the health-care bill

ISSN: 2167-1508
v5 n2240

Figure 2 Twitter Crowd sourcing of Opinions

ISSN: 2167-1508
v5 n2240

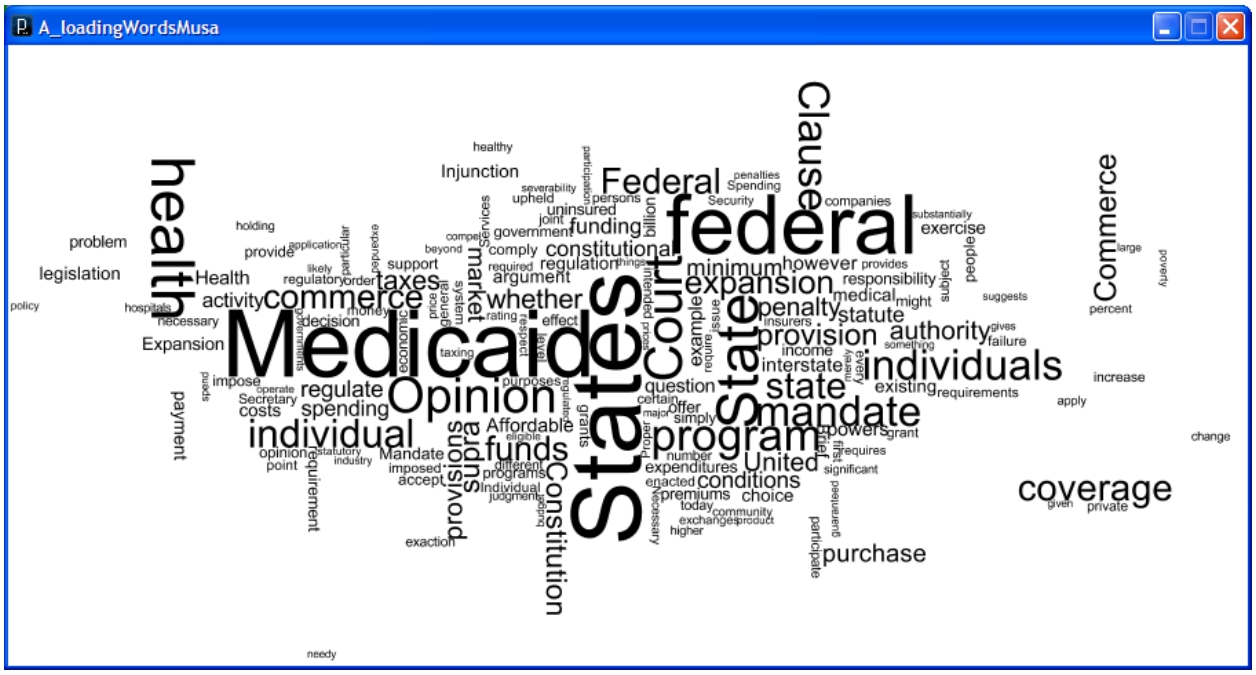

Figure 3 A wordCram word-cloud visualization of the supreme-court decision on the Affordable Care Act

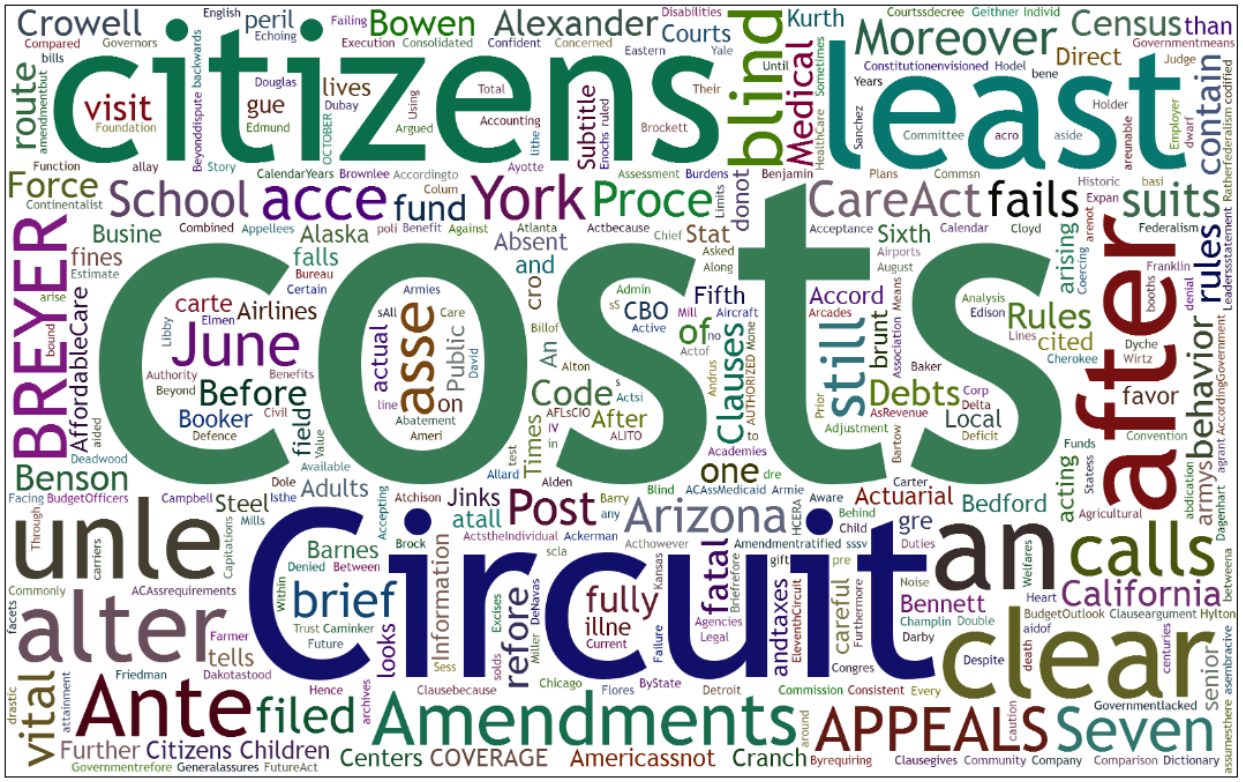

Figure 4 Our word-cloud visualization of the supreme-court decision on the Affordable Care Act

ISSN: 2167-1508
v5 n2240

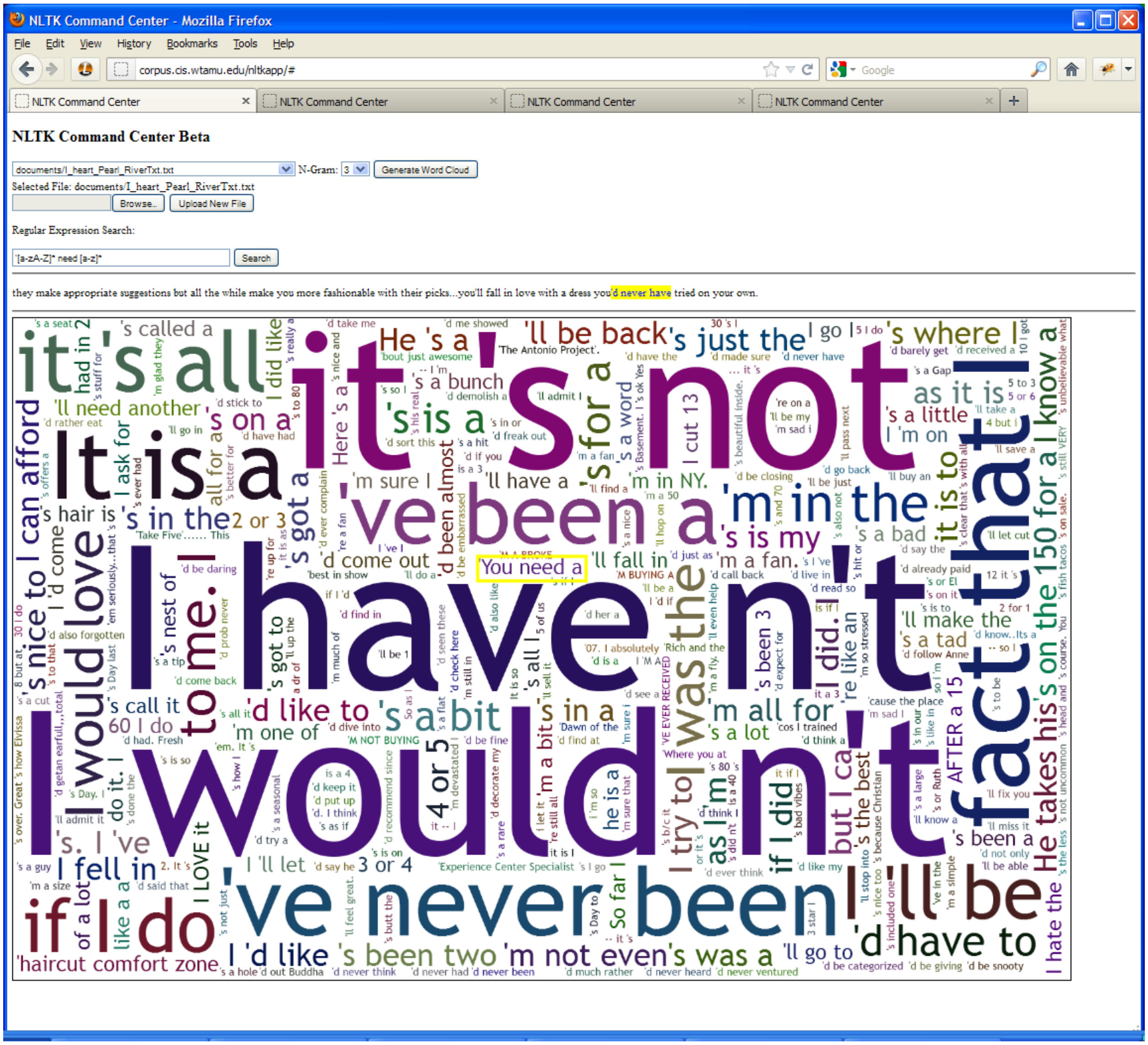

Figure 5 Control Center of Our Framework

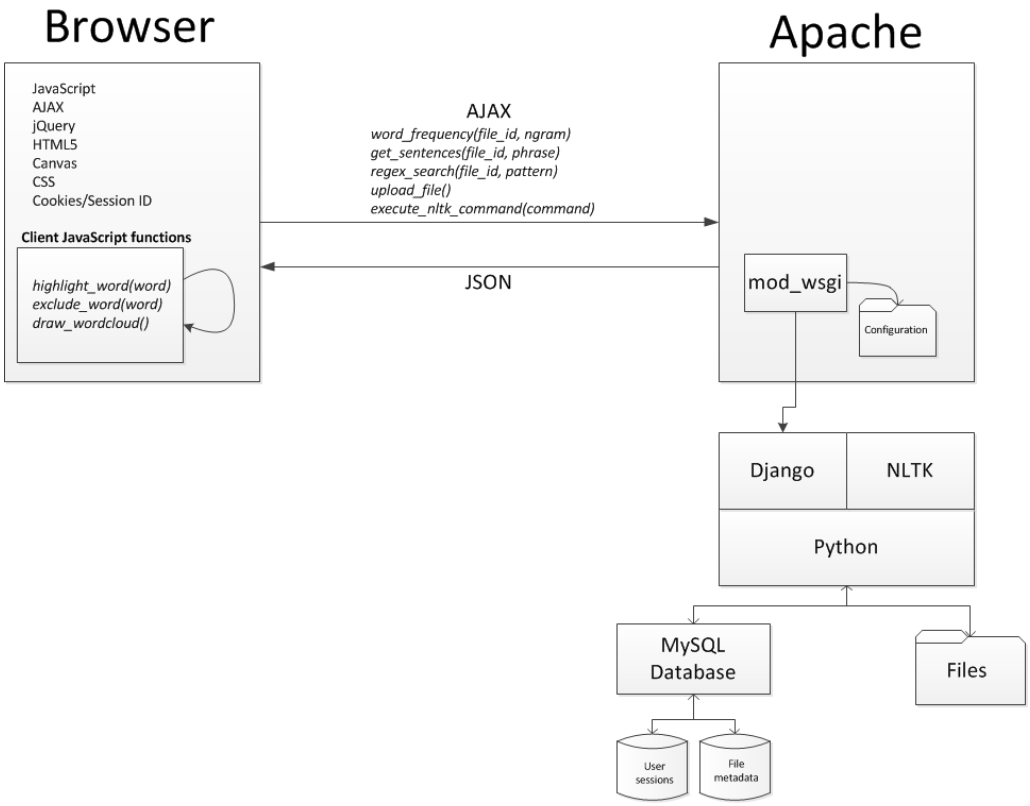

Figure 6 The Framework's Application Architecture