Decision-Making via Visual Analysis using the Natural Language Toolkit and R

Musa Jafar
mjafar@wtamu.edu

Jeffry S. Babb
jbabb@wtamu.edu

Kareem Dana
kdana@wtamu.edu

Computer Information and Decision Management
West Texas A&M University
Canyon, TX 79106 USA

Abstract

The preponderance of and rate of accumulation of textual data is now outstretching our ability to comprehend this text using conventional means. We extend our existing framework for the interactive visualization of textual data in digital format by including near-real-time text analysis using the R open source statistical and its analytical package(s). We utilize R as a pre-processor to programmatically gather and preprocess raw textual data generated by social media and incorporate it into textual corpora. The extended framework's back-end is a Django-based framework that relies on both the Natural Language Processing Toolkit (NLTK 2.0) and the R language and its rich set of packages. These tools are combined to present the user with a web-based and interactive n-gram/word cloud front end to visually and statistically analyze corpora built from our backend. We illustrate the use of this framework by utilizing the Twitter API to glean social trends that amount to visualizing zeitgeist. Our framework will allow subject-matter experts, typically in the humanities and social sciences, to develop alternative analyses of social phenomenon through text mining and visualization. The intent of our tool would be that subject-matter experts are able to manipulate text without the technical background in the tools typically used for these analyses, and without having to digest the entire works themselves, which is becoming impossible.

Keywords: Natural Language Processing, R, NLTK, Word cloud, social media, data visualization, corpus linguistics, n-gram.

1. INTRODUCTION

The hallmark of our progress towards the current information age has certainly been our ability to codify and normalize means of communication, both written and oral. As our civilizing has progressed, our written language persists as an artifact from which we can understand our culture, history, governance, civilization, and society. This is so as written language remains a primary means for transacting meaning, intent, values, and knowledge. In recent years, the volume of textual information that is generated by both humans and machines is so precipitously exponential that our means to analyze this data exceeds known human capacity (Farmer, 2010). The enormity of digitized text has transformed data mining, and the meta-phenomenon, known as Big Data, has become a necessity (Barlow, 2013; Jacobs, 2009; Zikopoulus, Eaton, deRoos, Deutsch, and Lapis, 2012).

Among the means of digesting Big Data is through data visualization supported by machine learning techniques. The visual analysis of plain text presents trends and also the caveats and quirks that have allowed humanity to diversify and coalesce (Norvig, 2009). What is remarkable is that the body of digitized text generated in the last 10 years has surpassed all of the textual knowledge recorded since the beginning of recorded history (Tynan, 2010). In general, we can soon expect that the sum of digitized text will be available for exploration, analysis, and reflection. It is supposed then that visually analyzing these texts, collectively and in the whole, can reveal trends not seen within in the details of any individual text. Moreover, if in the last 10 years we have been almost exclusively generating text in digital format, then roughly half of what we can analyze is very recent. When coupled with digitized historical text, we are able to develop data visualizations that span the human record. Google's effort to digitize over 15-million books and providing public domain frameworks to access those books is one such endeavor (Jean-Baptiste, 2011; Yuri, 2012).

One source of the new text being added to the overall textual record is that being generated by social media. These data are typically informal and represent, collectively, a zeitgeist for humanities and social scientists to examine and explore. The Oxford English Dictionary defines zeitgeist as: "the defining spirit or mood of a particular period of history as shown by the ideas and beliefs of the time." Despite hash tag metadata for categorization, data analysis tools, such as the interactive n-gram word cloud visualization framework presented in this paper, will better enable grasp of the zeitgeist that near-real-time analysis of social media textual data affords. Note that 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).

Recent phenomenon to which the zeitgeist of social media has been ascribed as being partially causal is the Arab Spring social upheaval in 2011 and 2012 in several Middle-Eastern countries (Howard et al., 2011). Many have ascribed the fluidity and feasibility of the Arab Spring as being due to the rapid availability of real-time information via social media channels.

Our objective is to provide a framework and tools where a non-technical researcher or practitioner, perhaps in the digital humanities (such as a journalist, a social scientist, 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 entire text?

2. RELATED WORK

In recent years, word clouds (sometimes referred to as tag clouds) have become a popular means to obtain the gist of a body of text. Such visualizations have been popularized in work of Viegas and Wattenberg (2007, 2009) in their *Many Eyes* and *Wordle* projects. These projects utilize manipulations of font-size, word placement, and word color to display the frequency distribution of words within a document. 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."

These projects, while inherently useful, are somewhat constrained by the minimal interactivity afforded in their presentation, such as the exclusion of words, modification of layout, and direction and color management are not within the users' control.

There remains a wide-spread appeal of word clouds where people have used them in many aspects of life. For instance, Sharma et al. (2012) use a word cloud approach to infer the social connections and networks (a 'who-is-who' approach) to highlight the characteristics of highly-regarded Twitter users by exploiting the metadata in the profiles of those users. The key words in their word cloud provide hyperlinks to more searches on Twitter.

In another case, Baroukh et al. (2011) used a "WordCram" (Oesper, 2011) word cloud to quickly summarize knowledge of biological terms. As WordCram is Processing-based (Processing being a Java library and framework for visualization), it is potentially not desirable as to a browser-based framework which relies on the HTML5 and 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) represent the relationship or co-occurrences between the entities to build word clouds of the nodes and their relationships.

In general, the aesthetics of presentation and the management of layout and placement of many existing word cloud frameworks are very appealing. In our framework, the aesthetic and layout issues continue to be addressed. However, our approach continues to work towards function and interaction with textual data aggregation and analysis as a primary concern; with the visualization optimizations following this. The advantage of our interactive n-gram word cloud approach is its increased interactivity with the word cloud to access its back-end natural language processing support. Our framework does not merely supply a static/semi-static web-page interface that heavily utilizes JavaScript or a standalone Java applet. Rather, our approach heavily relies on back-end processing, using both the Natural Language Toolkit (NLTK) and R, to analyze text beyond static presentation as a word cloud. Our framework provides a user interface to the NLTK package for the purpose of analyzing the text and inferring knowledge about it.

A major extension of our framework, developed to address an emergent need, is our use of R in addition to the NLTK, to examine near-real-time social media data, such as what can be obtained from Twitter. Social media (Twitter, Facebook, Google Plus, etc.) and news media (New York Times and others) companies are increasingly providing freed web-oriented Applications Programming Interfaces (APIs) access to their data. In recognition to the fact that we are generating new text data at a highly rapid pace, we recognize the need to retool our framework to accommodate the data available through these APIs. As the NLTK generally lacks deep statistical analytics, we have also incorporated R to assist in obtaining data from these APIs. Ultimately, our aim is to provide a portal for both subject-matter experts and laypersons to see the emergent zeitgeist inherent in our rich corpora of text.

3. OUR FRAMEWORK

Both the NLTK and R are at the center of our approach and framework. Recent additions to our framework include an ability to generate new corpora from a collection of separate files, to generate new corpora from the contents of a compressed Zip archive, and to provide more interactivity in the word cloud interface. The interface improvements increase interactivity by allowing users to select words for inclusion or exclusion and allowing for a side-by-side comparison between the modified and original word clouds.

We introduced R into our framework for its superior statistical and analytical capabilities. The contribution of the open source community to R made it the tool of choice for statistical and analytical data processing. The NLTK is a vital component as it is 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 the NLTK as a precursor for successful visualization as our visualization engine requires the outputs of computational linguistics. Furthermore, we require R to assist in analysis and to utilize its own extensive libraries and extensions to assist in obtaining social media data and building corpora of it.

Figure 1 shows the dashboard for our framework. Figure 5, 4, 5 and 6 are screenshots of the various modules of our framework. For the purpose of this paper, and to illustrate the current state of our framework, we used R and several R-packages (*twitteR*, *ROAuth*, *RJSONIO*) geo-locate the White House (President of the United States residence). We then requested 1000 tweets that have the word "white house" in them which originated within 5 miles of the White House. We then used the *tm*, *RWeka* and *Snowball* R-Packages to preprocess and clean the data and then generate a corpus based on the 1000 found tweets (Figure 7). While deeper analysis is possible within R, R is mainly services as a pre-processor for now. We then used our framework's administrative interface to load the 1000-Tweets corpus into the framework for further analysis.

With our framework, we can create, edit or delete corpora, and build a corpus by adding (deleting) new files into an existing corpus. Another means of upload is to obtain text data from a compressed Zip archive; such as those containing 1000 text files based on tweets pertaining to a certain subject or from a given user. The framework allows the user to analyze the whole corpus or a single document, by selecting the *generate word cloud* function of the framework. This creates a one-gram word cloud with the NLTK's point-wise mutual information (PMI) scoring method as the default. A two-gram or three-gram cloud can also be generated by the user along with various statistical measures such as t-tests, chi-squared tests, and others. For this transformation to occur, all documents are pre-processed in our framework using NLTK. Once tokenized into n-grams, we can use our visual interface to highlight a token. For instance, a user could click on the two-gram "geese honking" in the word cloud to see all the sentences of the token across the different documents encompassed by the corpus. We also provide a sorted list of all the tokens/frequencies as well as a regular expression search to find all of the token's usage in the body of text and the corresponding word-cloud grams.

What sets our approach apart from others we have examined is that it uses the word cloud as an interactive n-gram viewer for the purpose of text analytics. We do not only provide an n-gram word cloud, we also use the word cloud as an interactive analysis tool of the corpus. Thus, our Django-based web interface (both the administrative and user interfaces) serve as the front-end to an NLTK-driven (post processing) and R-driven (pre processing for now) back-end. After a document is rendered, we use AJAX to communicate with the 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 interaction with the visualization graphics directly.

4. OUR ARCHITECTURE

Figure 6 illustrates the architecture of our dashboard that connects extensively with the NLTK. 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 modern browser. It utilizes JavaScript, AJAX (Mozilla 2013), JSON (Mozilla 2013), jQuery (jQuery 2013), HTML5 (specifically the HTML5 Canvas and Local Storage features), and CSS. These latest web technologies enabled us to write a fully-featured, graphical, and interactive web application. We implemented an existing open-source HTML 5 Canvas-based word cloud library named *HTML 5 Word Cloud* written by Chien (2012).

The backend architecture is written in the Python programming language as the NLTK is a Python library. Accordingly, we used Django, which is a Python-based web application framework, as it can directly interface with the NLTK. For data persistence, we use a SQLite embedded database to store user session information and metadata regarding analyzed documents and corpora. The Django Framework provides a development web server to host and test Django applications during the development process. Our previous iteration of the architecture (Jafar, Babb, and Dana, 2012) utilized the Apache web server and MySQL database server. With this recent iteration, we now use the Django development web server and SQLite database. While Apache and MySQL are excellent tools, we have switched to the embedded web server and SQLite as both are lightweight in that they are easier to develop, test, and deploy. This switch has saved a lot of development time that would otherwise be spent maintaining and managing server infrastructure. Despite our entirely open-source implementation approach, our framework can be deployed on any other platform that Django supports, such as Windows and Internet Information Server (IIS).

By removing the dependencies of Apache and MySQL from the NLTK Command Center architecture, we have addressed previous design challenges of performance and scalability for our architecture. This is so as, in the current iteration of our framework; each user hosts an instance of the application on their client end. Such an approach paves the way to a flexible and extensible deployment solutions such as

Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) options. Accordingly, as it matures, we can refactor our architecture to support the Google App Engine or another cloud based service.

To facilitate our design goals, we added the ability for users to easily (through an intuitive web interface) create and modify entire corpora instead of just individual documents as was the case with our initial architecture. Corpora are stored directly on the file system and metadata including corpus name and contents are stored in the SQLite database. This feature allows users to combine similar documents into a single corpus and perform analysis.

Finally, our architecture now supports the vast lexical database of Princeton WordNet through the NLTK-WordNet interface (Miller 1995, Fellbaum 2005). WordNet provides part of speech tagging, synonyms, antonyms, definitions, and other lexical data that enhance the usefulness of our tool, especially when presented in a visual manner. Currently our interface with WordNet is rudimentary. We do envision a more robust interface for the purpose of word-sense-disambiguation, part-of-speech tagging and sentiment analysis and classification purposes.

5. IMPLEMENTATION DETAILS

When a user first connects to the NLTK Command Center website they are presented with a graphical user interface that allows them to perform various NLTK tasks (Figure 2). 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 the following features through a combination of JavaScript and AJAX-oriented Python methods which provide responsive access to the NLTK services in our framework:

- **Create Corpus** – Users of the NLTK Command Center can create a corpus (one or many) through the Django administrative interface. Text documents can be uploaded one at a time or many at once through a Zip archive. The user can add those documents to one or many corpora. This feature is usually the first step to using our application. Once a corpus is created and populated with texts, it can be visually analyzed by any of our other features.
- **Draw Word Cloud** – This is the main feature of our web application. One can select a document, multiple documents, or an entire corpus and draw a word cloud representation of those texts. Words or phrases that appear with the highest frequencies will appear larger in the word cloud. The word cloud is also interactive, so one can click on a word or phrase and get more information including the part-of-speech, definition, and frequency. Implementation wise, drawing the word cloud is broken down into two distinct steps: analyzing the text using the NLTK on the back end and rendering the word cloud itself in the user's web browser using HTML 5 and JavaScript.
- **Exclude Word Table** – When a user draws a word cloud, a table of all the words and phrases and their frequencies is also available. This table can be searched or sorted by word or frequency. Words can be excluded or explicitly included by using this table and the word cloud redrawn. Certain words may not be useful or interesting – this is where those words can be excluded all at once and only the words or phrases that the user is interested in can be shown in the word cloud. Clicking on a word in this table also highlights it in the word cloud making it very easy to find any word from any document.
- **Compare Word Clouds** – Using the same exclude word table, a user can fork multiple word clouds for side by side comparison. A new window will open in the web browser with the newly drawn word cloud so the user can compare multiple word clouds.
- **Highlight Words** – Being able to find words or phrases in the word cloud is a useful feature, especially if one draws a word cloud with hundreds of words. Using the exclude word table as describe above, one can sort the words or search for a specific word and then click on it to highlight that word in the word cloud. Clicking again will undo the highlight.

This is particularly useful for low frequency words that appear small in the word cloud but might still have some importance to the user or text in context.

- **Show Sentences** – This functionality 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 and looks for the 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.

  Being able to analyze multiple documents or an entire corpus means that a single word or phrase in the word cloud may actually appear in many sentences across multiple documents. We implemented a sentence-document map in Python that maps sentences to the document they are in and provide that information back to the user. Showing sentences will also show which documents those sentences are from.

- **WordNet Analysis** – In addition to showing the sentences a word appears in, right-clicking on a word (1gram support only) will also provide WordNet data about that word. Currently we display part-of-speech tagging, synonyms, and the definition.

In Figure 2, we illustrate general end-user workflow while using the framework. Users may select and/or upload their documents and render a word cloud from it. Interactivity allows for the user to exclude words and generate a modified word cloud which can then be compared to the original. If the text is new, it can be included an existing corpus or a new corpus can be generated. Textual analysis can be performed such that a sentence-document map is obtained. Analysis is performed against Princeton's WordNet lexical database for English (Felbaum, 2005). Though not yet implemented in our framework entirely, sentiment analysis is an area of great interest to us, especially the availability of social media data.

6. SUMMARY AND FUTURE WORK

With the rapid growth of textual data presaged by *Big Data*, we anticipate the need to perform textual analysis on a daily basis. We envision an end user who may lack the technical skills to issue hundreds of R and NLTK commands to perform these repetitive tasks. We present a framework that provides a layer of abstraction, and the underlying infrastructure, to facilitate these tasks. We utilized existing open source frameworks (NLTK, R, R-packages, Word Clouds, Django, etc.) to compound a framework that can address this need.

It is important to note that this architecture facilitates growth for semantic analysis by way of its use of R and packages. The interactive analysis also facilitates a visual sense-making for the user that sits between machine-learning semantic analysis and statistical summarization of the frequency of words.

In summary, we provide a powerful framework that allows a user with minimal technical experience to create, delete and modify corpora generated from digital text documents. The user is able to interactively analyze their corpora with an interactive n-gram word cloud viewer, include and exclude n-grams from their analysis, and drill down to the row sentences and their corresponding text documents. This technical approach is possible by utilizing R as a preprocessor and the NLTK for post processing and analysis.

With our interactive n-gram 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. Our premise rests within the context of the *Big Data* such that full comprehension of knowledge from texts will require information processing tools such as our framework. 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 continue to progress our framework with an aim to enable those with minimum computing knowledge to analyze digital text through interactive visualization. Our latest iteration seeks to not only serve as a decision support system, but to also develop a greater awareness of the meaning of these texts. 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. 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 the face of the root causes of the *Big Data* phenomenon, it is clear that our tools will need to take further analytical steps to tell the user what a body of text means. At the root of meaning is an aggregation and generalization of what other people think. In fact, much of the growth of textual data is opinion-rich: online communities, review sites, personal blogs, etc. (Pang and Lee, 2008). Our illustration using Twitter data is not an entirely new development and it makes clear the potential for the inclusion of sentiment analysis into our framework (Luce, 2012; Pak and Paroubek, 2010). While our emergent framework started as a generic tool for text representation, we realize the need for the tool's utility to grow such that it not only beautifies information, but also guides the user. Algorithmic approaches can render an initial portrayal of sentiment from which the end user, novice or expert, can then draw further conclusions. We have no doubt that attempts to enter into the foray of sentiment analysis will raise concerns for subjectivity (Liu, 2010), however we are interested in providing tools for others to make decisions.

Finally, our framework is designed to 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. Moreover, our framework is moving towards going beyond interactive visualization to performing sentiment analysis, and other *Big Data*-oriented tasks, in order to make the volume of data digestible. If successful, we envision our framework to be an Interactive web-based friendly layer on top of the Natural Language Processing toolkit (NLTK) that utilizes the power of R and R packages for the pre-processing of the raw data and for the analytical aspects of digital textual analysis.

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).

Barlow, M. (2013). Big data vs. Big reality. O'Reilly Radar. http://radar.oreilly.com/2013/06/big-data-vs-big-reality.html. Accessed June 28, 2013.

Howard, P.N., Duffy, A., Freelon, D., Hussain, M., Mari, W. & Mazaid, M. (2011). Opening Closed Regimes: What Was the Role of Social Media During the Arab Spring? Seattle: PIPTI. Retrieved June 28, 2013 from http://pitpi.org/index.php/2011/09/11/opening-closed-regimes-what-was-the-role-of-social-media-during-the-arab-spring/

Farmer, D. (2010). Study Projects Nearly 45-Fold Annual Data Growth by 2020. EMC Corporation. http://www.emc.com/about/news/press/2010/20100504-01.htm. Accessed June 28, 2013.

Fellbaum, Christiane (2005). WordNet and wordnets. In: Brown, Keith et al. (eds.), Encyclopedia of Language and Linguistics, Second Edition, Elsevier, Oxford, 665-670

Chien, T., G. (2012). HTML 5 Word Cloud. http://timc.idv.tw/wordcloud/en/. Accessed June 28, 2013.

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

Jafar, M., Babb, J., and Dana, K. (2012). A Framework for an Interactive Word-Cloud Approach for Visual Analysis of Digital Text using the Natural Language Toolkit. *2012 Proceedings of the Conference on Information Systems Applied Research*. November 1-4, 2012. pp. 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)*.

jQuery Foundation (2013) http://jquery.com/ Accessed Sept 13 2013

Liu, B. (2010). Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2, 568.

Luce, L. (2012). Twitter sentiment analysis using Python and NLTK. *Technical blog on web technologies*. http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/ Accessed July 03, 2013.

Miller, A. George (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

Mozzilla Developer Network (2013) AJAX https://developer.mozilla.org/en-US/docs/AJAX
Accesses Sept 13, 2013.

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

Oeper, 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).

Pak, A., & Paroubek, P. (2010, May). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In LREC.

Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and trends in information retrieval, 2(1-2), 1-135.

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.

Tynan, D. (2010). Prepare for Data Tsunami, Warns Google CEO. ITWorld, August 6, 2010. http://www.pcworld.com/article/202817/prepare_for_data_tsunami_warns_google_ceo.html. Accessed June 28, 2013.

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*.

Yuri L., et.al. (2012) Syntactic Annotations for the Google Books Ngram Corpus. Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (169-174).

Zikopoulos, Eaton, deRoos, Deutsch, and Lapis, (2012). Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data. McGraw Hill, New York.

Figures

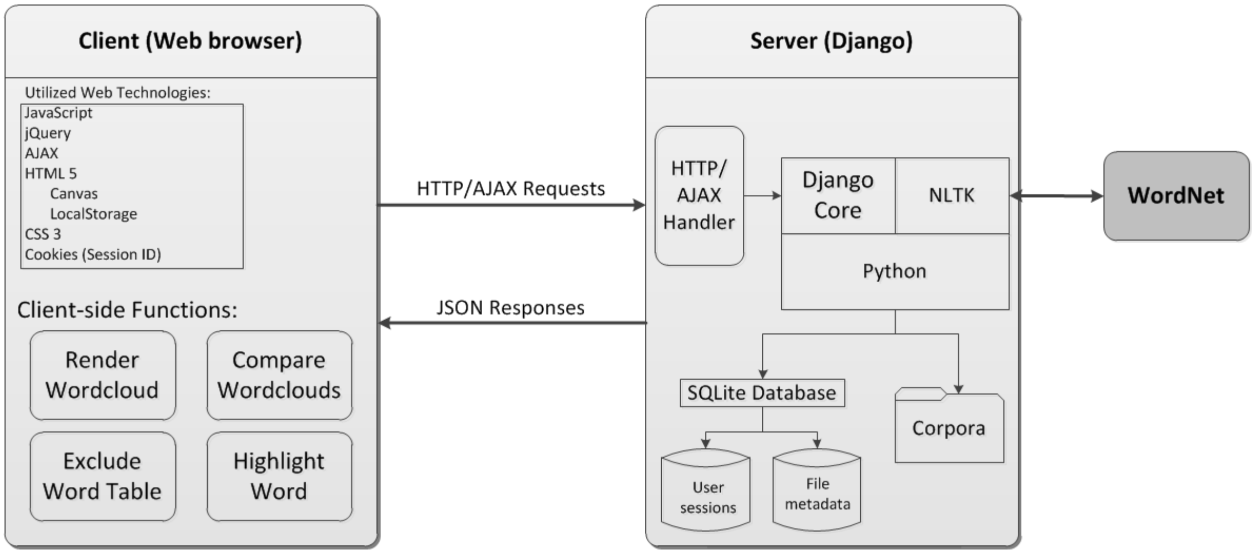


Figure 1 The Framework's Application Architecture

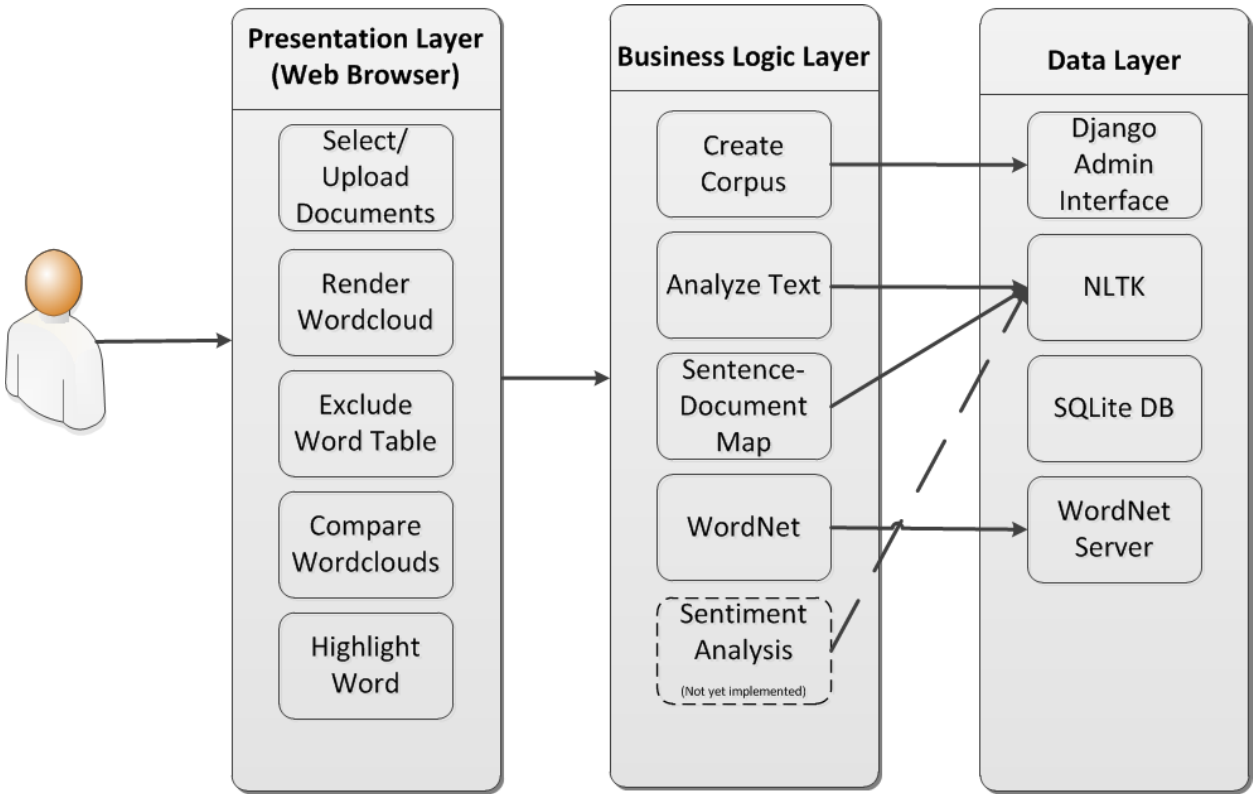
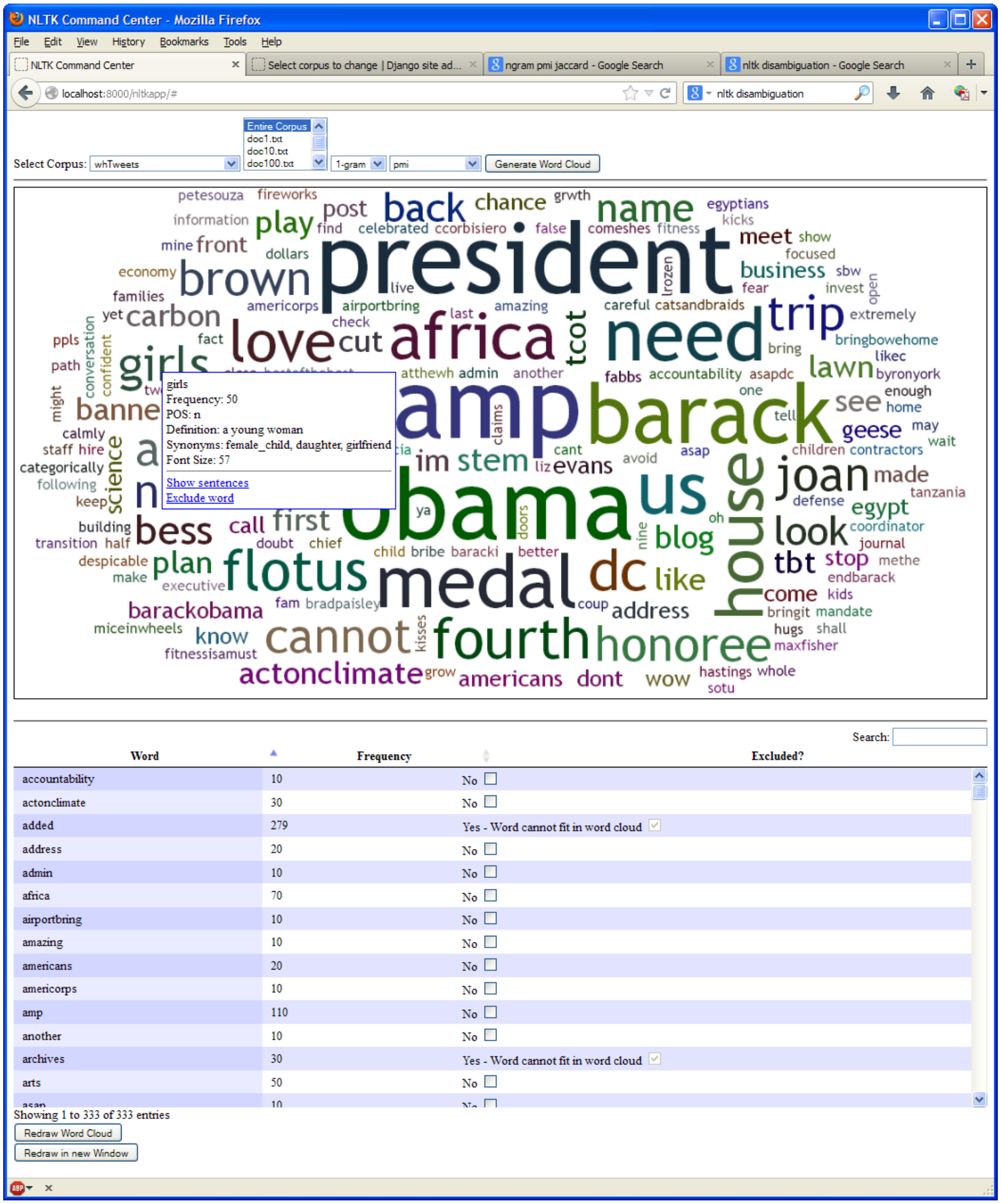


Figure 2 User Workflow through the Framework



Select Corpus: whTweets

1-gram pmi

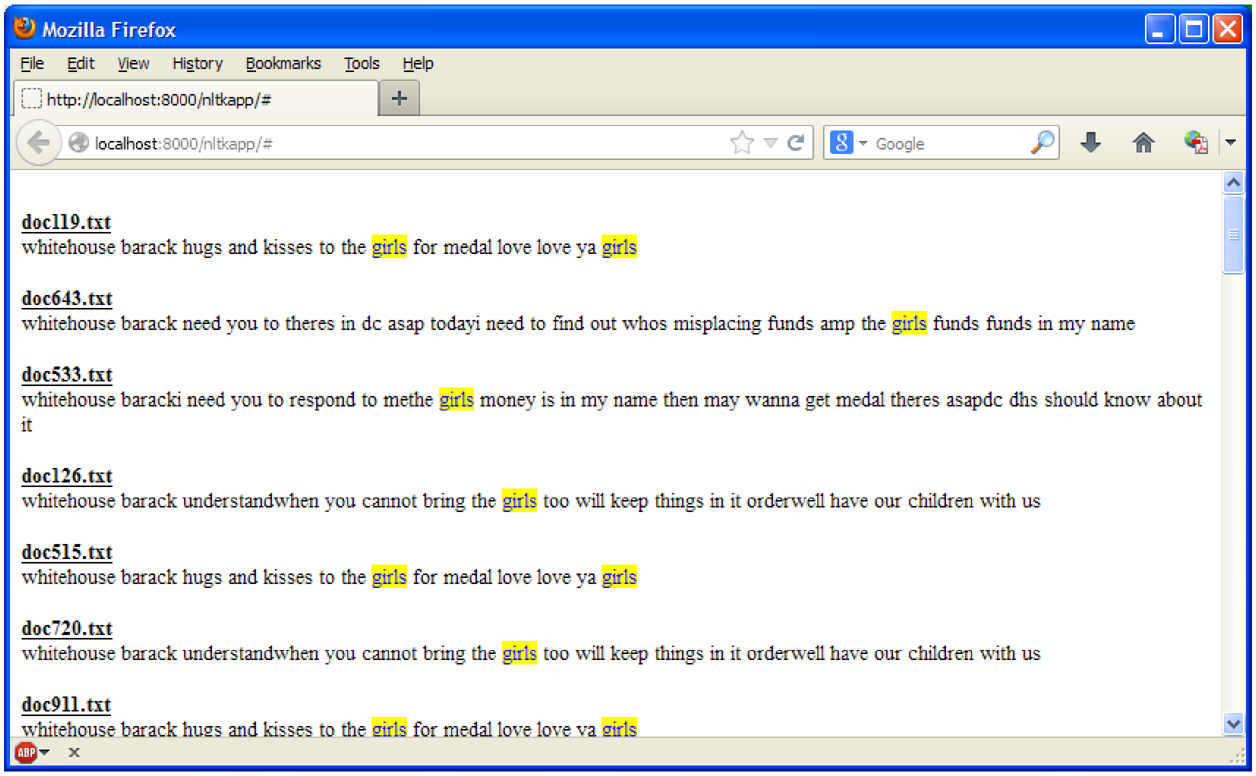
girls
Frequency: 50
POS: n
Definition: a young woman
Synonyms: female_child, daughter, girlfriend
Font Size: 57
Show sentences
Exclude word

Word	Frequency	Excluded?
accountability	10	No ☐
actonclimate	30	No ☐
added	279	Yes - Word cannot fit in word cloud ✓
address	20	No ☐
admin	10	No ☐
africa	70	No ☐
airportbring	10	No ☐
amazing	10	No ☐
americans	20	No ☐
americorps	10	No ☐
amp	110	No ☐
another	10	No ☐
archives	30	Yes - Word cannot fit in word cloud ✓
arts	50	No ☐
asap	10	No ☐

Showing 1 to 333 of 333 entries

Figure 3 An Interactive 1gram word cloud visualization of 1000 tweets within 5 miles off 1600 Pennsylvania Ave NW Washington, DC 20500

ISSN: 2167-1508
v6 n2821


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:8000/nltkapp/#
localhost:8000/nltkapp/#
Google

doc119.txt
whitehouse barack hugs and kisses to the girls for medal love love ya girls

doc643.txt
whitehouse barack need you to theres in dc asap todayi need to find out whos misplacing funds amp the girls funds funds in my name

doc533.txt
whitehouse baracki need you to respond to methe girls money is in my name then may wanna get medal theres asapdc dhs should know about it

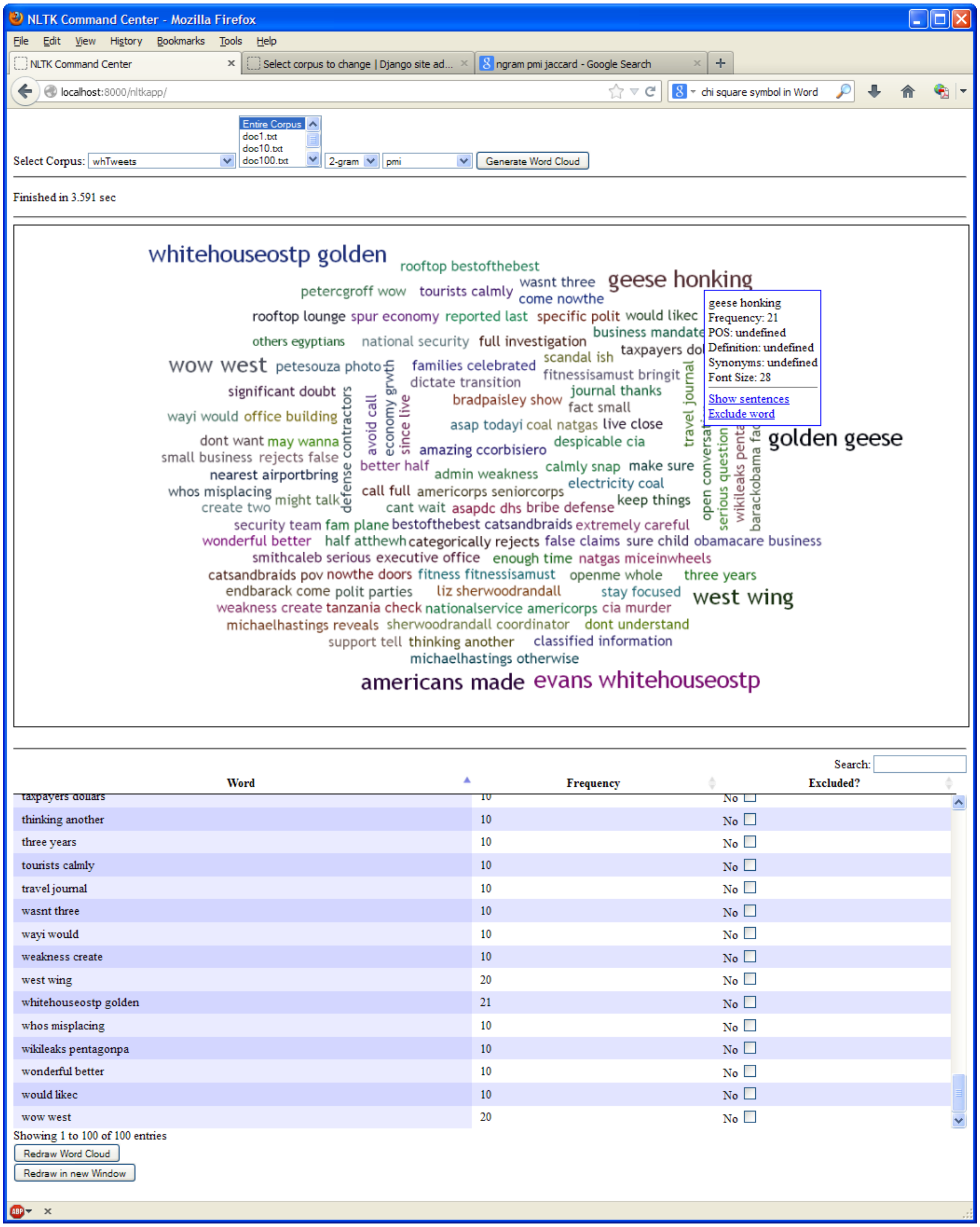
doc126.txt
whitehouse barack understandwhen you cannot bring the girls too will keep things in it orderwell have our children with us

doc515.txt
whitehouse barack hugs and kisses to the girls for medal love love ya girls

doc720.txt
whitehouse barack understandwhen you cannot bring the girls too will keep things in it orderwell have our children with us

doc911.txt
whitehouse barack hugs and kisses to the girls for medal love love ya girls

Figure 4 A Drill Down on girls from Figure 3 show the various tweets that contained the word



Select Corpus: whTweets

Entire Corpus
doc1.txt
doc10.txt
doc100.txt

2-gram

pmi

Generate Word Cloud

Finished in 3.591 sec

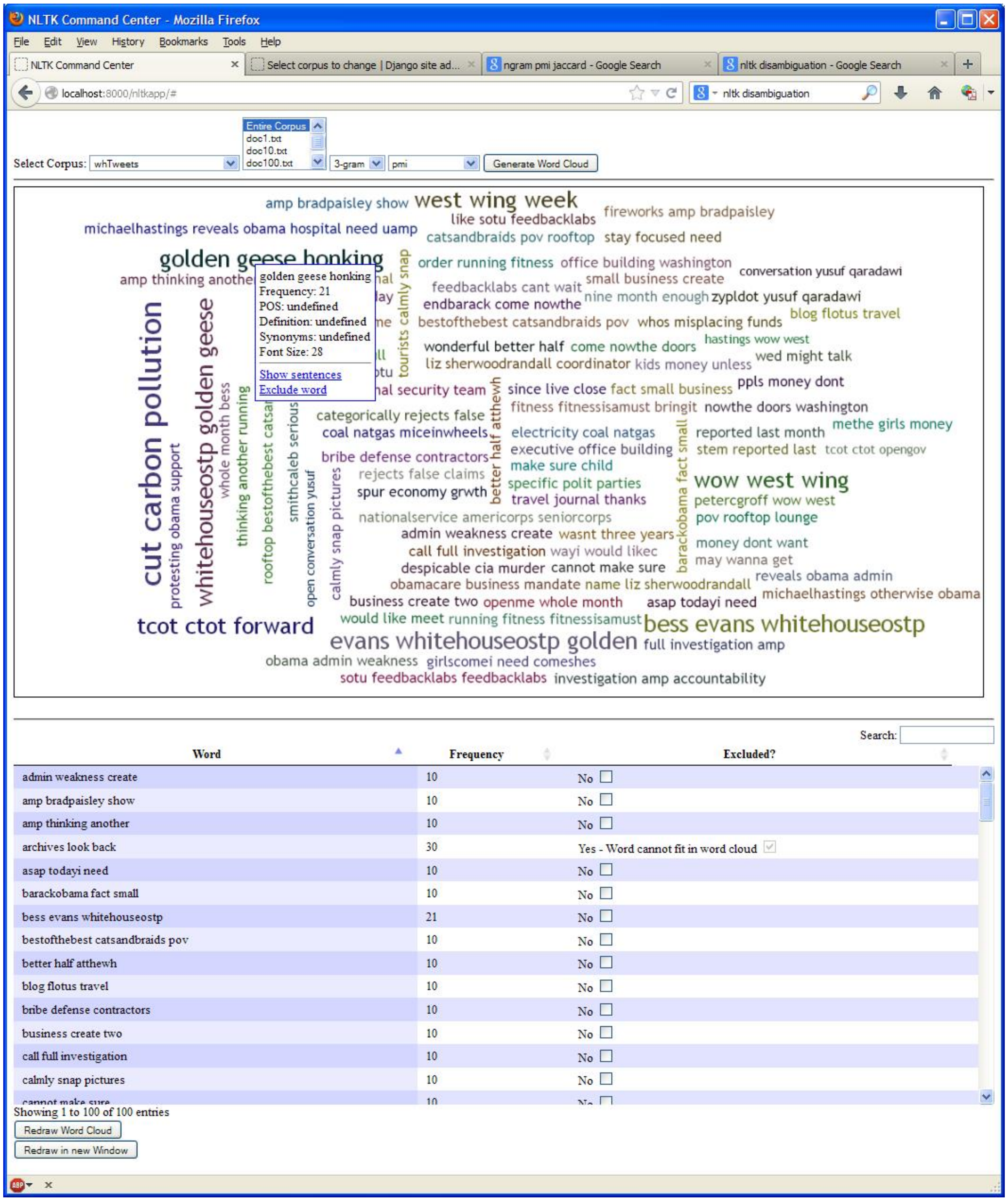
geese honking
Frequency: 21
POS: undefined
Definition: undefined
Synonyms: undefined
Font Size: 28
Show sentences
Exclude word

Search:

Word	Frequency	Excluded?
taxpayers dollars	10	No ☐
thinking another	10	No ☐
three years	10	No ☐
tourists calmly	10	No ☐
travel journal	10	No ☐
wasnt three	10	No ☐
wayi would	10	No ☐
weakness create	10	No ☐
west wing	20	No ☐
whitehouseostp golden	21	No ☐
whos misplacing	10	No ☐
wikileaks pentagonpa	10	No ☐
wonderful better	10	No ☐
would likec	10	No ☐
wow west	20	No ☐

Showing 1 to 100 of 100 entries

Figure 5 An Interactive 2gram word cloud visualization of 1000 tweets within 5 miles off 1600 Pennsylvania Ave NW Washington, DC 20500



Select Corpus: whTweets | Entire Corpus | 3-gram | pmi

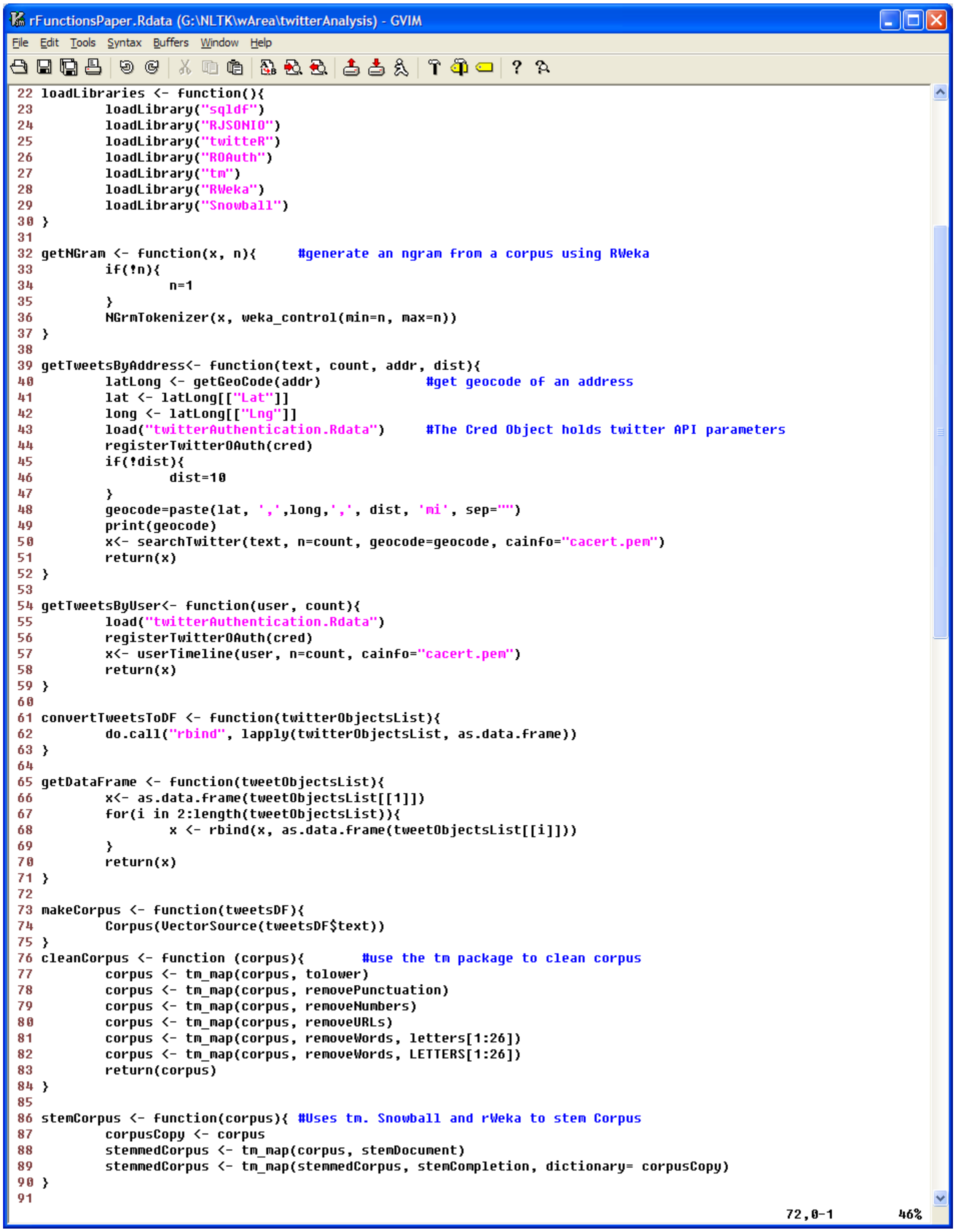
golden geese honking
Frequency: 21
POS: undefined
Definition: undefined
Synonyms: undefined
Font Size: 28
Show sentences
Exclude word

Search:

Word	Frequency	Excluded?
admin weakness create	10	No ☐
amp bradpaisley show	10	No ☐
amp thinking another	10	No ☐
archives look back	30	Yes - Word cannot fit in word cloud ✓
asap todayi need	10	No ☐
barackobama fact small	10	No ☐
bess evans whitehouseostp	21	No ☐
bestofthebest catsandbraids pov	10	No ☐
better half atthewh	10	No ☐
blog flotus travel	10	No ☐
bribe defense contractors	10	No ☐
business create two	10	No ☐
call full investigation	10	No ☐
calmly snap pictures	10	No ☐
cannot make sure	10	No ☐

Showing 1 to 100 of 100 entries

Figure 6 An Interactive 3gram word cloud visualization of 1000 tweets within 5 miles off 1600 Pennsylvania Ave NW Washington, DC 20500


rFunctionsPaper.Rdata (G:\NLTK\wArea\twitterAnalysis) - GVIM
File Edit Tools Syntax Buffers Window Help

```
loadLibraries <- function(){
        loadLibrary("sqldf")
        loadLibrary("RJSONIO")
        loadLibrary("twitteR")
        loadLibrary("ROAuth")
        loadLibrary("tm")
        loadLibrary("RWeka")
        loadLibrary("Snowball")
}

getNGram <- function(x, n){     #generate an ngram from a corpus using RWeka
        if(!n){
                n=1
        }
        NGrmTokenizer(x, weka_control(min=n, max=n))
}

getTweetsByAddress<- function(text, count, addr, dist){
        latLong <- getGeoCode(addr)             #get geocode of an address
        lat <- latLong[["Lat"]]
        long <- latLong[["Lng"]]
        load("twitterAuthentication.Rdata")     #The Cred Object holds twitter API parameters
        registerTwitterOAuth(cred)
        if(!dist){
                dist=10
        }
        geocode=paste(lat, ',',long,',', dist, 'mi', sep="")
        print(geocode)
        x<- searchTwitter(text, n=count, geocode=geocode, cainfo="cacert.pem")
        return(x)
}

getTweetsByUser<- function(user, count){
        load("twitterAuthentication.Rdata")
        registerTwitterOAuth(cred)
        x<- userTimeline(user, n=count, cainfo="cacert.pem")
        return(x)
}

convertTweetsToDF <- function(twitterObjectsList){
        do.call("rbind", lapply(twitterObjectsList, as.data.frame))
}

getDataFrame <- function(tweetObjectsList){
        x<- as.data.frame(tweetObjectsList[[1]])
        for(i in 2:length(tweetObjectsList)){
                x <- rbind(x, as.data.frame(tweetObjectsList[[i]]))
        }
        return(x)
}

makeCorpus <- function(tweetsDF){
        Corpus(VectorSource(tweetsDF$text))
}
cleanCorpus <- function (corpus){       #use the tm package to clean corpus
        corpus <- tm_map(corpus, tolower)
        corpus <- tm_map(corpus, removePunctuation)
        corpus <- tm_map(corpus, removeNumbers)
        corpus <- tm_map(corpus, removeURLs)
        corpus <- tm_map(corpus, removeWords, letters[1:26])
        corpus <- tm_map(corpus, removeWords, LETTERS[1:26])
        return(corpus)
}

stemCorpus <- function(corpus){ #Uses tm. Snowball and rWeka to stem Corpus
        corpusCopy <- corpus
        stemmedCorpus <- tm_map(corpus, stemDocument)
        stemmedCorpus <- tm_map(stemmedCorpus, stemCompletion, dictionary= corpusCopy)
}

                                                                    72,0-1        46%
```

Figure 7 Some of the R extraction and pre-processing code