

VizLinc: Integrating information extraction, search, graph analysis, and geo-location for the visual exploration of large data sets *

Joel C. Acevedo-Aviles, William M. Campbell, Daniel C. Halbert, Kara Greenfield
MIT Lincoln Laboratory, Human Language Technology Group, Lexington, MA, USA
{joel, wcampbell, daniel.halbert, kara.greenfield}@ll.mit.edu

ABSTRACT

In this demo paper we introduce *VizLinc*; an open-source software suite that integrates automatic information extraction, search, graph analysis, and geo-location for interactive visualization and exploration of large data sets. *VizLinc* helps users in: 1) understanding the type of information the data set under study might contain, 2) finding patterns and connections between entities, and 3) narrowing down the corpus to a small fraction of relevant documents that users can quickly read. We apply the tools offered by *VizLinc* to a subset of the New York Times Annotated Corpus and present use cases that demonstrate *VizLinc*'s search and visualization features.

Keywords

VizLinc, visualization, graph analysis, data exploration, information extraction, search, geo-location

1. INTRODUCTION

Information extraction refers to the task of automatically extracting structured information from unstructured documents. Sub-tasks like named entity, relationship, and terminology extraction are extremely useful to characterize the content of large text corpora and give data analysts a sense of what information might be present in such corpora. For many applications, characterizing individual documents is not enough. Linking relevant information across documents is the key to harnessing the informative power of a large homogeneous corpus.

In this paper we introduce *VizLinc*; an open-source software suite that integrates automatic information extraction, search, graph analysis, and geo-location for interactive visualization and exploration of large data sets. *VizLinc* helps

*This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

IDEA'14 August 24th, 2014 New York City, New York USA
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...\$15.00.

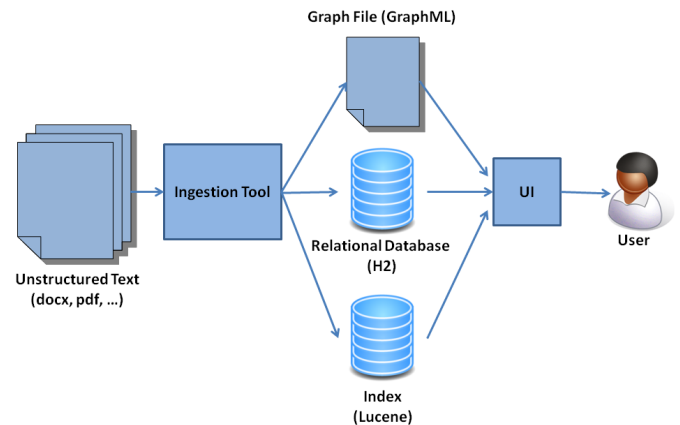


Figure 1: VizLinc main components. The Ingestion Tool processes text in a variety of formats and produces the metadata the UI requires as input for search and visualization.

users in: 1) understanding the type of information the data set under study might contain, 2) finding patterns and connections between entities, and 3) narrowing down the corpus to a small fraction of relevant documents that users can quickly read. *VizLinc* is self-contained, does not require connections to online components, and scales to thousands of documents. All software is publicly available through GitHub.¹

The rest of this paper is organized as follows. Sections 2 and 3 present an overview of the main components of the *VizLinc* software suite and the technology used to implement it. Sections 4 to 6, offer a look into *VizLinc*'s features and usage. Lastly, we apply our techniques to a data set and present use cases to demonstrate *VizLinc*'s capabilities in section 7.

2. SYSTEM OVERVIEW

VizLinc is a software suite composed of two main applications: the *Ingestion Tool* and the *User Interface (UI)*. The Ingestion Tool takes a set of documents as input, extracts information from unstructured text, and stores the extracted information in the format that the UI needs to allow users to search, visualize, and explore the documents' content. The

¹<https://github.com/mitll/vizlinc>
https://github.com/mitll/vizlinc_db
https://github.com/mitll/vizlinc_ingester

process of converting the input documents into information-rich data structures, or simply the *ingestion process*, will be covered in detail in section 3. For now, it suffices to say that data ingestion is carried out entirely by the Ingestion Tool in two major stages: *Information Extraction* and *Metadata Generation*. During Information Extraction, mentions of people, locations, and organizations are identified in text. Once identified, locations are geo-coded and people are linked based on their co-occurrence patterns in the data set. In addition, the text is indexed for search and retrieval. The metadata generation step takes the extracted information and stores it in an H2² relational database, a GraphML³ graph file, and a Lucene⁴ index. The UI takes the database, graph file, and index as inputs and presents a graphical user interface for interactive visualization and exploration of the original data set. For the rest of this paper, we will refer to the UI simply as *VizLinc*. Figure 1 depicts the interaction between the aforementioned components.

The Ingestion Tool is written in the Groovy programming language. *Groovy* is an agile and dynamic language for the Java Virtual Machine⁵. The graph database we use for data ingestion has a robust implementation in Groovy thus making it an ideal choice for the Ingestion Tool. For ease of use, we have developed a graphical user interface in Java Swing that calls the appropriate Groovy classes as needed.

As the reader probably inferred by this point, the UI is written in the Java programming language. Specifically, the UI is a Gephi⁶ plugin. *Gephi* is an interactive visualization and exploration platform for large graphs [1] and powers all graph-related features in the UI. Gephi, in turn, is based on the *Netbeans Platform*⁷, a generic framework for rapid development of Java Swing applications. One of the key distinctions of software built upon the NetBeans Platform is modularity [2]. This distinction made the integration of VizLinc’s UI with Gephi a seamless one. For the rest of this paper, we will refer to the UI component simply as VizLinc.

3. DATA INGESTION

Data ingestion refers to the sequence of processing steps that generate the necessary metadata for later visualization and exploration in VizLinc. Figure 2 shows what these steps are and the order in which they are executed.

VizLinc admits text in a variety of formats including Microsoft Office formats (.docx, .doc, .xls, ...), Portable Document Format (PDF) and HyperText Markup Language (HTML).⁸ For this reason, the first step in the ingestion pipeline is extracting the text contained in the input documents. We use the tools provided by *Apache Tika*⁹ for this purpose. Other content, such as images, is ignored.

Once text is extracted, we perform named entity recognition on each document using the *Stanford NER* [3] recognizer. During this step named entities, specifically people, locations, and organizations, are identified and extracted.

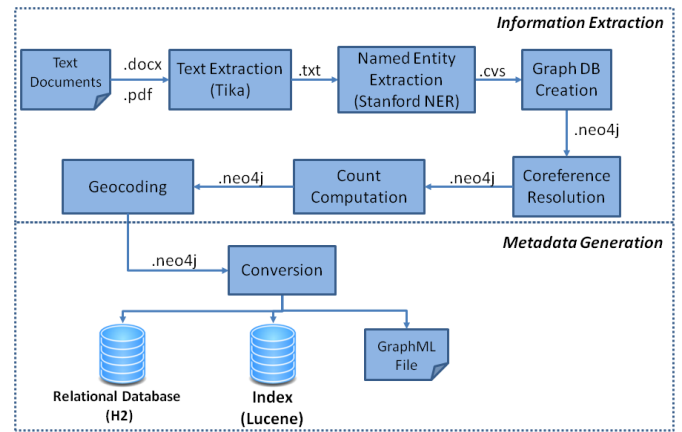


Figure 2: The data ingestion process.

Each instance of an entity in a text is called a *mention*. All mentions, information about the documents in which they appear, and their positions within those documents are stored in a *Neo4j*¹⁰ graph database. The need to store and retrieve links between the metadata that will be generated in subsequent steps, makes this data representation an intuitive and efficient one for our purposes [6]. The graph database is augmented as ingestion progresses and, by the end of the pipeline, stores the results of all the steps performed during this process.

Mentions can have different forms yet refer to the same entity. For instance, the person entity *John Fitzgerald Kennedy* might be referred to as “John F. Kennedy”, “Kennedy”, and “JFK”. VizLinc aims at discovering interesting patterns in text by unveiling connections between the entities mentioned. To this end, it is critical to find all the mentions of an entity both within a document and across documents in the corpus. The task of finding all expressions that refer to the same entity is denominated *coreference resolution* and is the goal of the fourth step in the ingestion pipeline. In VizLinc, approximate string matching and a simple set of rules are brought together to: 1) find all mentions of an entity within a document (e.g., “John F. Kennedy”, “Kennedy”) and assign them a single canonical form (e.g., “John F. Kennedy”) that is then used to 2) link all mentions of the same entity across documents.

Ingestion proceeds by calculating the number of times each entity is mentioned in the entire data set and the number of documents in which each appears. These values are stored in the graph database for later presentation in the graphical user interface.

One of VizLinc’s main features is a map that displays the locations mentioned in a selected subset of the document set under analysis. To render this possible, locations have to be resolved to latitude/longitude coordinates that can be then highlighted in the map. This is precisely what the *Geocoding* step depicted in figure 2 does. Internally, we have used a number of approaches to carry out this step for our data sets of interest. In the version that we have publicly released, the user can point the Ingestion Tool to an online geocoding server. The *Geocoding* step marks the end of the Information Extraction stage.

²<http://www.h2database.com/>

³<http://graphml.graphdrawing.org/>

⁴<http://lucene.apache.org/>

⁵<http://groovy.codehaus.org/>

⁶<http://gephi.org/>

⁷<https://netbeans.org/features/platform/>

⁸For a list of all supported formats, see <https://tika.apache.org/1.4/formats.html>.

⁹<http://tika.apache.org/>

¹⁰<http://www.neo4j.org/>

During the Metadata Generation stage, particular data structures are created and saved to disk. Most of the meta-data generated is stored in an *H2* database. *H2* is an open source database engine written in the Java programming language². Its speed, ability to run without a server, and seamless integration with Java applications were the main reasons why we chose it over other database engines.

Co-occurrences of person entities in documents are encoded in the form of edges between nodes of a graph. For that reason, we store this information in a GraphML file. *GraphML* is a comprehensive file format for graphs which consists of a language core to describe the structural properties of a graph and a flexible extension mechanism to add application-specific data³. Each node in the graph represents a person entity mentioned in the data set. An edge exists between a pair of nodes if the corresponding entities co-occur in more than two documents. Co-occurrence is a symmetric relation therefore edges in the graph are undirected. In sections 5.5 and 7 we will discuss how this co-occurrence network can be used to find coherent groups and “important” people [4].

Lastly, *Lucene* is used to generate an index that stores the entire text content of the input documents in a format suitable for string searching. *Apache Lucene* is a high-performance, full-featured text search engine library written entirely in Java. *Lucene* is an open source project available for free download⁴.

4. VIZLINC INPUT

When run for the first time, VizLinc prompts the user for the system paths of the database, index, and graph file generated by the Ingestion Tool. Additionally, VizLinc requires a tile source to populate its map. Two tile source types are supported in the current version. If users have pre-generated tiles and saved them as images, the directory in which they were saved can be specified. Otherwise, an HTTP map server can be specified through a URL. Once the input is specified and loaded, users can visualize and explore their data sets. At any point, users can point VizLinc to a different data set or tile source.

5. DATA CHARACTERIZATION

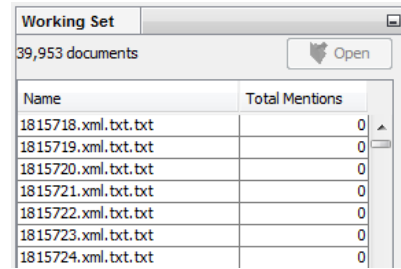
Upon loading our data in VizLinc we can immediately get a sense of the composition of our text corpus. Figure 3 shows a snapshot of the UI’s main components or *views*. In the following sub-sections we describe each of these views and how they can be used for data exploration.

5.1 Working Document Set

The *working set* is the set of documents currently being visualized in VizLinc. At first, this set consists of all documents in the data set but as search queries are applied, this set gets narrowed down to a relevant subset of the corpus (see section 6). Keep in mind that one of the main goals of VizLinc is to empower users to quickly filter out those documents that might not contain relevant information.

The *Working Document Set* view lists the specific documents that are part of the working set. This view is shown in figure 4.

At any point in time, the number shown across the top of the view represents how many documents are being analyzed and represented in all views. The entries under the *Total*



Name	Total Mentions
1815718.xml.txt.txt	0
1815719.xml.txt.txt	0
1815720.xml.txt.txt	0
1815721.xml.txt.txt	0
1815722.xml.txt.txt	0
1815723.xml.txt.txt	0
1815724.xml.txt.txt	0

Figure 4: Working Set view

Mentions column will be explained in section 6.

5.2 Document Viewer

Ultimately, users should be able to easily read the informative sections of a document, as determined by the search query, and draw relevant conclusions. Selecting a document name in the *Working Set* view and clicking on the *Open* button will show the document’s content in the *Document Viewer*. This view displays the text extracted from the selected document in its raw form. All formatting information, other than capitalization and spacing, is discarded in the ingestion process. Figure 5 shows this view.

If the *Highlight All* check box is selected, the Document Viewer highlights all the mentions found in the document. A color code is used to distinguish between people, locations, and organizations. If this check box is not selected only those mentions that match the search query are highlighted.

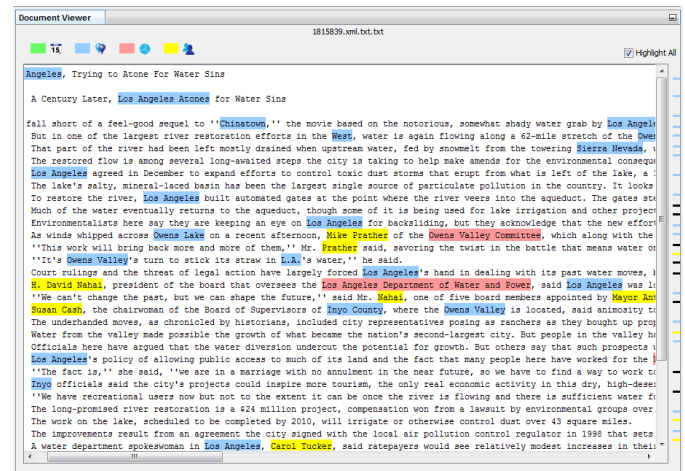


Figure 5: Document Viewer

5.3 Search View

Figure 6 shows the Search View. As the name implies the *Search* view allows users to search for particular terms or entities in the data set. However, that is not its sole utility. This view also lists all the people, locations, and organizations automatically extracted during the ingestion process. The number that appears next to the entity type is the number of entities of that type present in the working set. Each entity is shown along with its mention and document count. The mention count is the number of times an entity is referred to in the working set whereas the document count

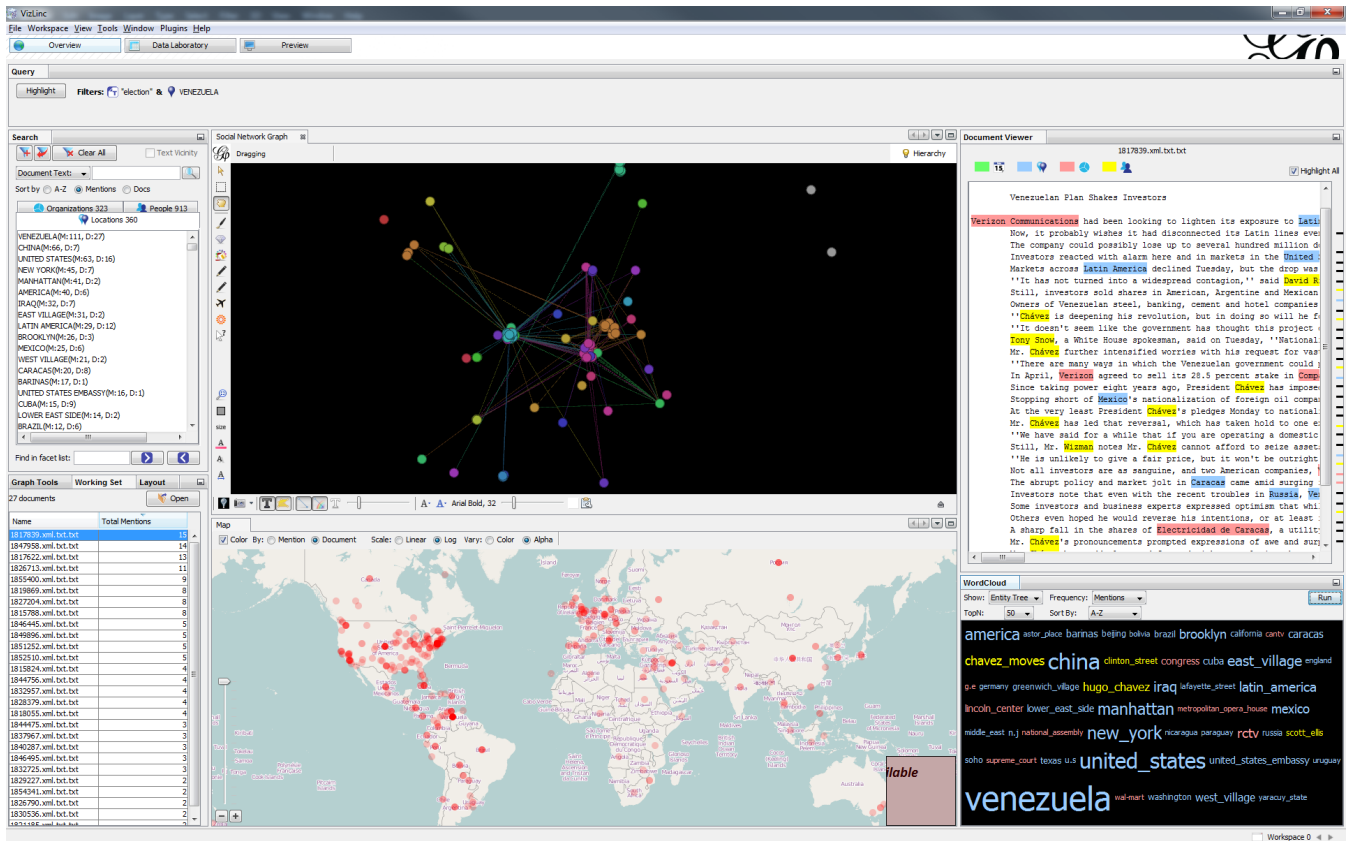


Figure 3: VizLinc: user interface

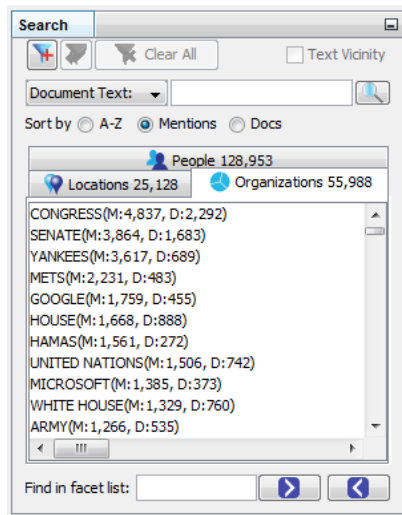


Figure 6: Search view. Here we show the most mentioned organizations in the New York Times dataset as extracted by VizLinc’s Ingestion Tool.

is the number of documents in which an entity is mentioned. Both counts are shown in this view only as it pertains to the current working set, i.e., for all the documents that match the current query. If there is no query the counts correspond to the whole data set. All lists can be sorted alphabetically, by decreasing mention count, or decreasing document count. The search-related features of this view will be covered in section 6. Figure 6 shows the list of organizations extracted from the New York Times data set (see section 7.1) sorted by mention count.

5.4 Map

The *Map* view places the locations present in the working set on a geographic map. A small circular waypoint is drawn for each location. Users can navigate the map by zooming and panning. The color or alpha value of each circle can represent either the mention or the document frequency of the corresponding location. Adjusting the alpha value of waypoints based on frequency is particularly useful when there are a large number of locations in the working set. The most frequent locations become clearly visible whereas locations with few mentions/documents fade into the background.

5.5 Graph

The *Graph* view shows the co-occurrence network of all the people mentioned in the working set. Nodes in the graph represent person entities and edges represent document co-occurrence between the linked entities. This is a direct visualization of the graph generated during data ingestion.

VizLinc contains all of Gephi’s visualization, analysis, and exploration capabilities. In addition, we have made some useful graph analytics accessible through the *Graph Tools* view. The following sections describe those analytics.

Node Centrality

The centrality of a node measures its relative importance within a graph. In the context of VizLinc, centrality can be an indicator of how important a person is in the social structures described in the working set. We have included

two centrality metrics: Eigenvector Centrality and PageRank[5]. Both metrics are based on the concept that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. Users can choose to represent the nodes’ centrality score through modifying their size and/or color.

Clustering

The clustering feature groups related nodes in a graph and colors them accordingly. Clustering is based on the *InfoMap* algorithm which attempts to find community structure based on the flow of information in the graph.[7]

N-Hop Network

Highlighting a node in the graph and clicking on the *1-Hop Network* button displays a network consisting of the selected node, all of its neighbors, and all the edges that exist between them. Note that this graph would not necessarily contain all the people in the working document set as normally as the “seed” node does not need to be one of the terms in the search query. Similarly, clicking on the *2-Hop Network* button would generate and display a graph containing all the neighbors of the nodes in the 1-hop network and the edges between all of them.

5.6 Word Cloud

The Word Cloud provides an aggregated view of the most frequent entities. The canonical names of the *N* most frequent entities are laid out in a grid and their font size is adjusted so that it is proportional to their mention or document count.

6. SEARCH

So far, we have discussed how VizLinc can be used to ingest documents containing text, summarize the entities mentioned in those documents, and visualize their co-occurrence patterns and geographical placement. In this section we will talk about how VizLinc can direct users to relevant sections of a document through searching.

A search query acts as a document filter and can contain one or more terms. Documents that match all the terms in the query are kept in the working set whereas those that don’t are eliminated. As a result, all views are updated to display only those entities present in the new working set. Narrowing down the working set and updating all views accordingly constitutes the basis of discovering patterns and relevant information in VizLinc.

Not only can users target a sub-set of all documents but they can also quickly navigate to the sections within those documents where the target entities are mentioned. This is done by opening one of the documents in the working set. The Document Viewer then shows the content of that document and highlights all the instances of the query terms using different colors for each term type. In addition, color-coded markers for each line containing a match are shown along the right side of the Document Viewer for quick access to the relevant document sections (see figure 8).

In the following sections, we discuss the two search features VizLinc offers.

6.1 String Search

String search refers to the process of finding all the documents that contain a specific string. *Lucene* is used both

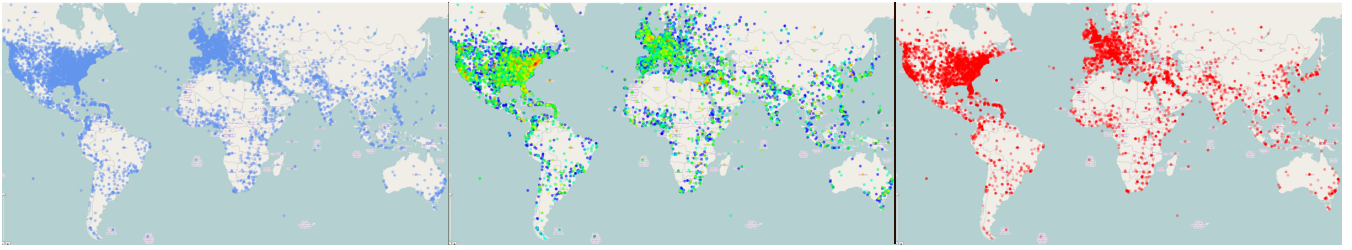


Figure 7: Map showing three different parameter settings: a waypoint per location (left); color scale representing the location’s mention count, where blue correspond to the lowest value and red to the highest (center); and alpha value representing mention count, where the highest frequency locations are rendered solid and the lowest frequency locations are not drawn.

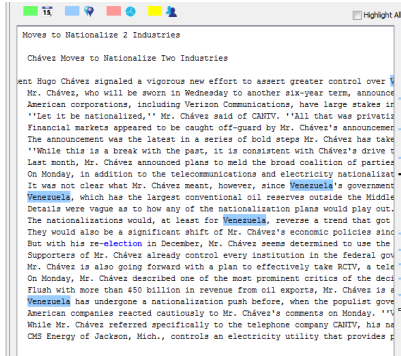


Figure 8: Document content showing the matches to the query String:elections & Location:Venezuela

for indexing the text during ingestion and to search the documents through VizLinc. Matching documents become the working set and all views are updated to reflect the entities mentioned in them. The nature of this type of search implies that its results could mixed documents referring to different entities. For instance a search for the string “washington” will return documents mentioning Washington D.C. (location), George Washington (person), and Washington State University (organization).

6.2 Entity Search

An entity-based search query contains one or more named entities (i.e., locations, persons, and organizations) and returns all the documents that mention those entities. This type of search differs from string search in that a query entity could resolve to several different strings if they are all different ways to mention the same entity. This is the result of within/across document co-referencing during ingestion. For instance, a search for Person:John M. Smith might resolve to mentions “John Smith”, “John”, or “Mr. Smith”. This feature might represent a significant advantage over string searching if users have a particular entity in mind. For instance, if we are interested in those documents that mention the state of Washington, a search for Location:Washington will exclude documents that mention President Washington and not the location.

There are many ways to execute an entity search in VizLinc. In the Search view, users can select an entity from the list and click the *Add Filter* button on the view’s toolbar. Alternatively, users can drag the entity name from the list

and drop it in the *Query* view. Entities can also be added to a query from the graph, map or word cloud by right-clicking on their representation (node, waypoint, or label, respectively) and selecting *Add to Query* from the context menu.

7. CASE STUDY

In this section, we present a few hypothetical use cases based on the results obtained by processing and analyzing real data with VizLinc. These use cases and results should give readers some insight about the type of patterns and information VizLinc can reveal. All hypotheses drawn from our visualization and stated in this section were later confirmed by examining the content of the appropriate documents.

7.1 New York Times Articles (2007)

The New York Times Annotated Corpus [8] contains over 1.8 million articles written and published by the New York Times between January 1, 1987 and June 19, 2007 with article metadata. The data set we processed is a subset composed of all articles written in the year 2007. This subset is composed of 39,953 documents containing thousands of entities.

Figure 9 shows the map, graph, and word cloud for the whole New York Times 2007 data set. The map shows that most location mentions are concentrated in the U.S.A. and Western Europe. It is hard to make sense of the graph when it contains so many nodes. Upon closer examination thanks to VizLinc’s graph navigation and clustering features, we can see that clusters belong to different categories. Politicians, artists and sports personalities all have their own clusters. The word cloud shows the 50 most salient terms in the data set. Not surprisingly, locations New York, United States, New York City, Iraq, Manhattan and Washington are heavily mentioned. Organizations like Congress, Senate, Yankees (New York Yankees), and Google also form part of the list of most mentioned entities.

We will rely on a hypothetical use case and potential action path to illustrate VizLinc search capabilities on the New York Times data set. Let us say that we are interested in elections around the world. A first approach would be to do a search for the string “elections”. The working set decreases from nearly 40,000 to 834 documents that contain that string. The list of documents can now be sorted by the total number of mentions and we could browse the contents of the top hits. Instead, we will take a look at the map and

location list to see what locations co-occur the most with the term “elections”. The reader should keep in mind that, after executing a query, all views are updated to show different visualizations of the content of the matching document set only i.e., the *new working set*. The entity list in the *Search* view shows that “Iraq”, “United States”, and “Israel” are the most mentioned locations in conjunction with “elections”. Examining the map shows activity in many other parts of the world including the major countries in South America. Let us say that Venezuela piques our interest, so we add **Location:Venezuela** to the query from the map view.

The working set now contains 12 documents that could be browsed within minutes if so desired. The resulting graph shows the people mentioned in these documents and it is much more suitable for visual analysis than the original one.

To get a sense of the importance of each individual in the working set as described by the co-occurrence relation defined in previous sections, we re-size the nodes in the graph according to their centrality score. Also, we can cluster this new sub-graph to reveal any community structures present. Figure 10 shows part of the resulting graph.

The graph suggests that one of the most central people is Hugo Chavez. Hugo Chavez was the president of Venezuela in 2007 and had been re-elected the previous year. This is not new information but it demonstrates VizLinc’s ability to find central people with respect to some user-defined context. Clustering resulted in three major communities; a subset is shown in Figure 10. Upon examination, it can be noticed that the three clusters illustrated group three different types of actors: USA political and media figures (top-left), South American political figures (top-right) and artists (bottom-left). From this point on, if we were interested in the sentiment and opinions of U.S. politicians towards the government of Venezuela we could add members of that community to the query. If what is relevant to us is stories about South American leaders and the government of Venezuela we would add members of the second community to my query and examine the resulting documents. Authors and entertainers appear in the graph due to spurious co-occurrences in articles that contain lists spanning a variety of unrelated topics.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we have introduced VizLinc and described how it combines information extraction, graph analysis, and geo-location for visualization and exploration of text corpora. We have also presented a case study, centered on a compilation of articles from the New York Times, to demonstrate VizLinc’s features.

Now that we have achieved our principal goal of creating a complete framework for data ingestion, visualization, and exploration, our future work will focus on making each component more generic and robust. Modules such as the ones that generate the graph and perform coreference resolution, yielded reasonable results on the data for which VizLinc was initially intended. However, these modules turned out to be rather simplistic for most of the text genres we have tested so far. Multiple-term searches could also be improved by restricting the distance at which both terms can appear in a document. This will avoid documents in which terms co-occur but are in fact unrelated. Expanding queries to support “and” and “or” operations is also a subject for future work. Finally, we understand that user-defined algorithms

and entity types will be required to analyze certain data sets efficiently. Therefore we would like to include a mechanism that would allow users to add these custom components with ease.

9. REFERENCES

- [1] M. Bastian, S. Heymann, M. Jacomy, et al. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 8:361–362, 2009.
- [2] T. Boudreau, J. Tulach, and R. Unger. Decoupled design: building applications on the netbeans platform. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*, pages 631–631. ACM, 2006.
- [3] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics, 2005.
- [4] A. Özgür, B. Cetin, and H. Bingol. Co-occurrence network of reuters news. *International Journal of Modern Physics C*, 19(05):689–702, 2008.
- [5] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. 1999.
- [6] M. A. Rodriguez and P. Neubauer. The graph traversal pattern. *arXiv preprint arXiv:1004.1001*, 2010.
- [7] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. In *Proceedings of the National Academy of Sciences*, page 1118, 2001.
- [8] E. Sandhaus. The new york times annotated corpus ldc2008t19. *Linguistic Data Consortium*, 2008.

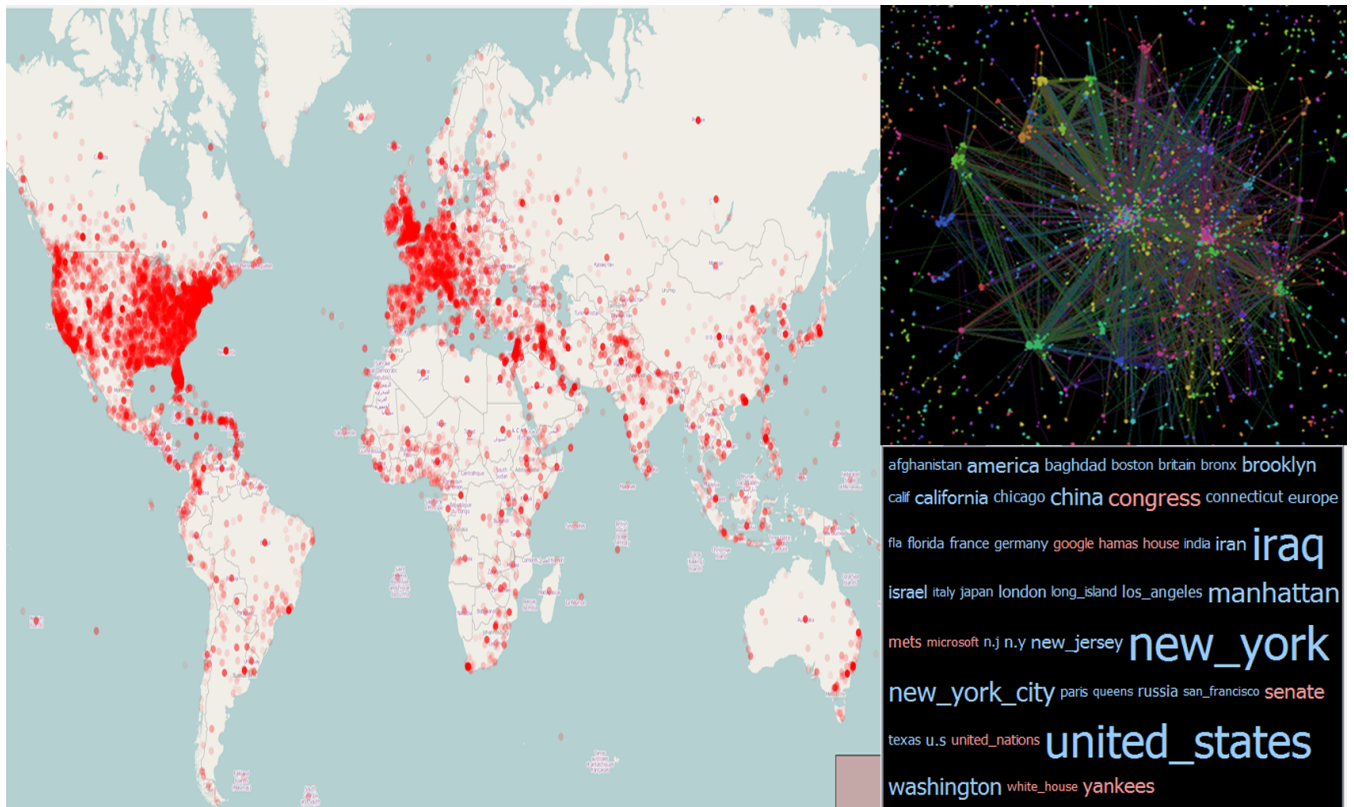


Figure 9: Map, co-occurrence graph, and word cloud for the New York Times 2007 data set.



Figure 10: Section of the graph after executing the query String:elections & Location:Venezuela. The nodes have been re-sized according to their centrality scores and clustering has been run on this sub-graph. Clusters shown group three types of people: USA political figures (top-left), South American political figures (top-right) and artists (bottom-left).