Journal of Computer Information Systems

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ucis20

Story Analysis Using Natural Language Processing and Interactive Dashboards

Michel Mitri

To cite this article: Michel Mitri (2020): Story Analysis Using Natural Language Processing and Interactive Dashboards, Journal of Computer Information Systems, DOI: 10.1080/08874417.2020.1774442

To link to this article: https://doi.org/10.1080/08874417.2020.1774442

Published online: 02 Jul 2020.

JOURNAL OF COMPUTER INFORMATION SYSTEMS
https://doi.org/10.1080/08874417.2020.1774442

Story Analysis Using Natural Language Processing and Interactive Dashboards

Michel Mitri

James Madison University, Harrisonburg, Virginia, USA

ABSTRACT

This paper discusses Story Analyzer, which uses a natural language processing (NLP) library and sophisticated data visualization libraries to produce dashboards of interrelated and user-responsive visualizations depicting actors and their interactions in a textual narrative, along with locations, times, and other contexts. Story Analyzer performs information extraction using Stanford's CoreNLP's NLP services including sentence recognition, tokenizing, parts-of-speech identification, dependency parsing, named entity recognition, coreference resolution, and temporal tagging. Visualization is done through D3 and scalable vector graphics (SVG) which provide powerful control over gelements and shapes in browser-based user interfaces. Google Charts and Maps are also used for visualizations. Development using NLP for unstructured textual data involves challenges, limitations, and ambiguities that distinguishes it from applications using structured data. Therefore, the paper also discusses issues and limitations inherent with using NLP libraries, and presents workarounds when applied to story analysis. Story Analyzer is applied to a contemporary news article regarding data privacy issues.

KEYWORDS

Natural language processing (NLP); data visualization; story analysis; NLP accuracy issues and mitigations; story dashboard generation

Introduction

Story Analyzer is a software application that helps users visualize and understand a story through the use of natural language processing (NLP) and data visualization APIs. Here, the term "story" refers to a narrative that involves people, groups, or other entities (subjects) performing actions that can affect other people, or entities (objects). These events occur in certain places and at certain times, and they may include other contextual features of interest. The term "story analysis" pertains to identifying these key elements of the story (people, groups, subjects, objects, actions, time, place, and other contexts), and moreover to represent the relationships between these elements for each action that takes place in the story. The software described in this paper attempts to visually and interactively answer this question: Who did what to whom, where, and when did it happen, and what else was going on at the time?

Stories can be fictional or non-fictional. These run the gamut from novels, novellas, or short stories to newspaper or magazine articles, to historical reports, and even to personal reminiscences and memoirs. In the context of information systems, stories can also include user stories, which form the basis of requirements elicitation during a systems analysis process. Legal and political documents are also often highly narrative in form.

Note that this type of "text understanding" is very different from other kinds. Reading and understanding a story is not the same as reading and understanding a technical manual or an anatomy textbook. NLP can be applied to many types of reading tasks, but for this paper, we focus on story analysis, and especially the idea of subjects performing actions that impact objects at certain places and times, and under certain contexts.

There is a well-known saying: "A picture tells a thousand words." Visualization software can be used to draw a picture of this narrative via a variety of graphics (e.g. word clouds, chord diagrams, force graphs, timelines, maps, etc.) to show the interaction between subject, object, actions, places, times, and other contexts. These visualizations, leveraging on the NLP functionality introduced above, can help a user quickly capture the essence of a story.

NLP involves a blend of artificial intelligence, computer science, machine learning, and computational linguistics. NLP systems perform many tasks necessary for making sense of text or speech recognition. Some of these are grammatically focused, such as parts-of-speech (POS) tagging and syntactic parsing. Others are based on recognizing co-occurrences of entities in a document (coreference resolution), recognizing named entities, and interpreting temporal expressions. At a deeper level, NLP forms a venue for attempting to infer the underlying meaning of text; this has historically been termed "natural language understanding".[12]

A key element of natural language understanding, as applied to the story analysis task described in this paper, is information extraction (IE), which can be defined as "automatic extraction of structured information such as entities, relationships between entities, and attributes describing entities from unstructured sources such as text corpus or text documents".[3] IE involves several tasks relevant to story analysis, including named entity recognition (NER), relation extraction, event extraction, temporal expression, and template filling.[4] IE is related to information retrieval (IR), but differs in an important respect.[5] Information retrieval is used to identify relevant documents, a task often associated with search engines like Google. By

CONTACT Michel Mitri mitrimx@jmu.edu Computer Information Systems and Business Analytics Department, MSC 0202 James Madison University, Harrisonburg, VA 22807

contrast, information extraction uses NLP techniques to take the retrieved unstructured text data from documents and impose structure and "meaning" onto it. Although IR is an important means of filtering out irrelevant text from a myriad of documents, it is outside the scope of this paper. IE is more pertinent for the task of helping users to quickly understand the underlying meaning of the text once retrieved.

NLP and information extraction for story analysis

Story analysis and understanding is an important area of AI and NLP research, and has a rich history in artificial intelligence going back almost a half-century.[1,2,6] Since those early days, the NLP discipline has advanced to the point that there are powerful open source APIs available, such as Apache's OpenNLP,[7] software tools from the Berkeley Natural Language Processing group,[8] and Stanford's CoreNLP annotators.[9] These APIs include models for performing many important NLP and information extraction tasks. Some models are generated through machine learning algorithms, trained on large quantities of annotated text documents. Others are more rule-based, often utilizing linguistic and grammatical rules. In addition to Java and Python APIs, a growing number of cloud service providers include NLP functionality, such as Amazon's AWS Comprehend, Microsoft's Azure Cognitive Services, and IBM's Watson.

These APIs and cloud services provide a variety of useful NLP functions that can assist with story understanding. Some are particularly relevant to Story Analyzer, including the following:

- Breaking a text document into individual sentences (sentence splitting)
- Tokenizing a sentence (breaking it into individual "words")
- Identifying parts of speech (POS) within a sentence (nouns, verbs, adjectives, adverbs, etc.). Typically based on the Penn Treebank[10]
- Named entity recognition – recognizing names of people, places, organizations, dates/times, etc.
- Constituency parsing – constructing taxonomies of noun phrases and verb phrases of a sentence
- Dependency parsing – constructing the graph of dependency relationships between terms in a sentence
- Co-reference resolution – finding all expressions that refer to the same entity in a text
- Temporal tagging – recognizing and normalizing temporal expressions (e.g. "next Wednesday", "the previous summer")

Academic literature of recent years includes many descriptions of NLP-related applications, which use information extraction techniques in a variety of domains. One example is Content Analyzer and Information Extraction System (CAINES), applied to analyzing the content of online reviews[3] and terrorism incident reports.[11] CAINES evolved from Flexible Information extRaction SysTem (FIRST) which was applied to analyzing textual NASDAQ financial information.[12] In addition to NLP techniques, CAINES was also combined with the Text Analysis and Mining System (TAMS) to analyze reports from health care organizations to evaluate the environmental impacts of their supply chain practices.[13]

Another application is Analysis of Language and Content in a Digital Environment (ALCIDE),[14] a web-based platform using a variety of NLP information extraction techniques to assist humanities scholars analyze literary and history documents. Visual Narrator[15] applies NLP to the task of software requirements elicitation, analyzing user stories to generate OWL-based ontologies. Ceren et al.[16] describe an NLP application for detecting stories and themes embedded in longer online postings of extremist groups. Zhu et al[17] apply information extraction to online news analysis. Chen et al[18] describe an application that uses NLP techniques for automatic grading of student essays.

Each of these applications makes use of one or more of the NLP functions listed above, and many use other statistical features such as colocation analysis, topic modeling, keyphrase extraction, and other statistical approaches. Most of these applications involve some aspects of story analysis, particularly the identification of actions that take place, the subjects that generate those actions, and the objects affected by the actions. Some also make use of ontologies, such as the lexical database provided by Princeton's WordNet.[19]

Visualizations for story analysis

The user experience of a story analysis application can be greatly enhanced via data visualization tools that operate on the NLP results. This experience enables users to more rapidly understand a body of text, compared to the task of meticulously reading a full narrative or exhaustively searching through the graphs and relationships of NLP output. Consequently, many applications involving NLP tasks also include visualizations to enhance the user experience.

Word clouds (also called tag clouds) are ubiquitous in text analysis applications, and for good reason. First, each word or phrase in a word cloud can include many visual cues for conveying various meanings depending on the application. Visual cues for a word cloud include font size, color, text shadows, bold/italic/underline, and blinking or animated text. Second, each word or phrase in a cloud is an individual interactive element, which can respond to various user actions (e.g. mouse click, hover, drag, etc.). Third, when combining multiple word clouds together, or combining word clouds with other visualizations, this provides a rich tapestry of possibilities for showing a picture of a story. Consequently, there have been many innovative uses of word clouds in software involving NLP, information retrieval, or other text analytic approaches.

ALCIDE[14] combines word clouds with maps, timelines, and network graphs for various visual depictions of literary and historical text analysis. Zhu's news analysis system[17] uses these visualizations as well. Koch et al[20] describe VarifocalReader, a multilayer visualization that hierarchically displays chapters, then sections, then word clouds of text content. Word Cloud Explorer[21] involves a central interactive word cloud; selecting a word from the cloud brings up other panel views and filters. ReCloud[22] uses word clouds for analyzing online reviews.

One example of combining multiple word clouds is ConcentriCloud[23] a visualization composed of an inner cloud surrounded by layers of surrounding clouds. ConcentriCloud is useful for showing a contrastive view of multiple documents. The outermost layer will contain information from individual documents. Interior layers contain intersections of words from their connected exterior clouds, and the innermost contains words that are in the intersection of all document text. This allows users to see what words or phrases are shared between which documents. Another approach to multi-cloud visualization is the use of parallel tag clouds (PTC).[24] Here, words are arranged parallel columns, one for each distinct subset of the textual data; the authors show an application that contrasts court decisions from various U.S. circuit courts.

NLP applications can make use of other visualizations as well. Typically, stories involve a timeline, and although the narrative may jump forward and backward through time, it is useful to see a linear sequence of events through time. Timeline visualizations are useful for this task.[25–27] Stories also depict relationships between people, organizations, or other entities. Chord visualizations can be useful for showing interactions between pairs of these entities.

Story Analyzer, the application described in this paper, similarly combines multiple clouds and other visualizations for assisting users to see the big picture of a story. The following section describes Story Analyzer's architecture. This includes the use of Stanford's CoreNLP[9] to extract key elements and relationships of a narrative, and the D3 JavaScript API[28,29] to generate interactive and interrelated visualizations for showing a picture of the story.

Story analyzer's use of coreNLP to identify story elements

Story Analyzer is a Java application using Stanford's CoreNLP for information extraction (IE). The application automatically generates a web page with D3 word clouds and other visualizations, and includes interactive JavaScript functionality that assists users to quickly navigate and discern the key aspects of a story that is input to the system.

Story Analyzer contains two main components: information extraction and visualization. The information extraction module makes extensive use of CoreNLP, as outlined in Figure 1. Sentence splitting, tokenizing, and POS tagging are very straightforward tasks and CoreNLP is very accurate with these functions. Named entity recognition and parsing are highly useful tasks but are considerably less accurate in NLP's current state of the art, with accuracy rates in the 80%+ range. Coreference resolution, which is vital for connecting related themes in a story, is far less accurate. 60% accuracy is considered a good score in the current NLP climate. These limitations impact the accuracy of Story Analyzer's results and are discussed in a subsequent section of the article.

Story Analyzer's IE algorithm, written in Java, relies on POS tagging for identifying the actions of a story. For each sentence in the story, the verbs are identified, and each verb becomes the starting point for a dependency graph navigation to identify the subject, objects, places, times, and other contexts of the action. An action involves verbs, but an action is not necessarily a single verb in isolation. For example, a verb may be negated, as in "The dog did not bite the mailman". Or, consider this sentence: "The police failed to apprehend the suspect". Here we have two verbs, "fail" and "apprehend", and the action includes both. Actions may involve adverbs, such as "anxiously waiting". If you want to capture the meaning of a story, then these nuances should be considered. So, although the verb gives the start of identifying the action, often more is required, some of which involve additional POS tags, as well as navigation through the sentence's dependency graph.

When an action is identified, the information extraction module searches the dependency graph for subjects and objects of the action. A dependency graph is created via CoreNLP's dependency parsing annotator. Dependency parsing of a sentence generates a set of binary relations between

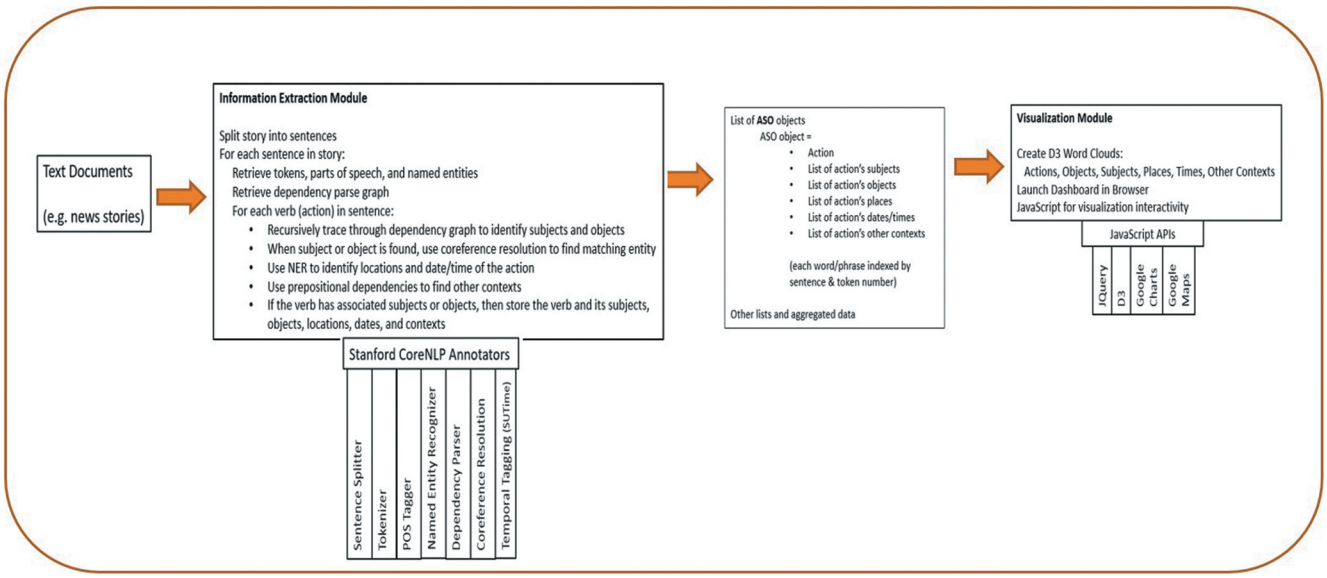

Figure 1. Story analyzer's system architecture.

two words in the sentence, called dependency relations.[30] A dependency relation is a binary predicate involving two words, a governor and a dependent.

Two key dependency relations of an active voice sentence are the nominal subject (nsubj) and the direct object (dobj). If the same verb is the governor of both nsubj and dobj relations of the same sentence, then the nsubj dependent is the subject of that verb and the dobj dependent is its object. Similarly, passive voice sentences will involve an agent relation and a passive nominal subject (nsubjpass) relation.

For example, consider the sentence "The dog bit the mailman." Also, consider the sentence "The mailman was bit by the dog." Both are depicted in Figure 2.

CoreNLP's dependency parser recognizes over fifty different types of dependency relations.[30] A sentence typically contains several dependency relations between words. Some are particularly useful in identifying the connection between subjects and objects of complex sentences, or for more completely describing a subject or object. Of course, nsubj, dobj, agent, and nsubjpass, are needed for identifying the direct subjects and objects of an action. But others provide additional support in complex sentences where the subject or object is only indirectly linked to the verb or may not be linked via these relations but still qualify as a subject or object of the action. These include relative clause modifiers (rcmod), adjectival modifiers (amod), noun modifiers (nn), prepositional modifiers (prep), verb modifiers (vmod), and open clausal complements (xcomp). The full set of relations in a sentence forms a graph, and the process of connecting subjects to objects involves a recursive traversal of this graph[31] focusing on the most relevant dependencies. Use of Semgrex,[32] a query language for dependency graphs, assists with this effort.

When a subject or object is found via nsubj, dobj, agent, or nsubjpass dependencies, is this subject/object an entity that was previously discussed in the text? For example, the subject or object could be a name or a person; did that person come up before? Or, it could be a pronoun; to whom or what does this pronoun refer?

To answer this question, Story Analyzer uses CoreNLP's coreference resolution annotator. As mentioned earlier, coreference resolution is far from perfect; current tests give it only 50–60% accuracy rates, and there is a heavy tradeoff between precision (avoiding false positives) and recall (avoiding false negatives). Nevertheless, coreference resolution is essential for producing a coherent understanding of a story when the same subject or object is referenced many times using different labels throughout the text.

CoreNLP has three versions of coreference resolution (neural network, statistical, and deterministic), of which the neural network is the most accurate (F1 score 60) but also the slowest. This is the one used by Story Analyzer. Coreference resolution of a text document results in a set of coreference chains, each consisting of a set of mentions. Each mention of a chain is a phrase in a sentence and is identified by the sentence number and the token start and end numbers within the sentence. Additional information about the mention include whether it is animate or inanimate, and a gender specification (male, female, neutral). All mentions in a chain are supposed to refer to the same entity or theme, but the imperfections of coreference resolution can lead to missing a coreference (lack of recall) or incorrectly assuming a coreference (lack of precision). We'll see some of these oddities using a sample story in the next section, and later discuss workarounds for mitigating coreference precision or recall errors.

Using these CoreNLP annotators, the IE algorithm produces a set of Action-Subject-Object (ASO) instances. Each individual action that is either associated with a subject or associated with an object generates a unique ASO instance. The ASO instance includes the action (including with token and sentence numbers), along with five lists, containing the associated subjects, objects, places, times, and other contexts, respectively. The algorithm also maintains frequency counts (after coreference resolution) for each of these entities, which helps depict which entities predominate in the story. This data becomes the input to Story Analyzer's visualization module.

The visualization module generates the HTML and JavaScript for creating and displaying six interactive word clouds. The elements of the ASO list facilitate the display of relationships between subjects, actions, objects, places, times,

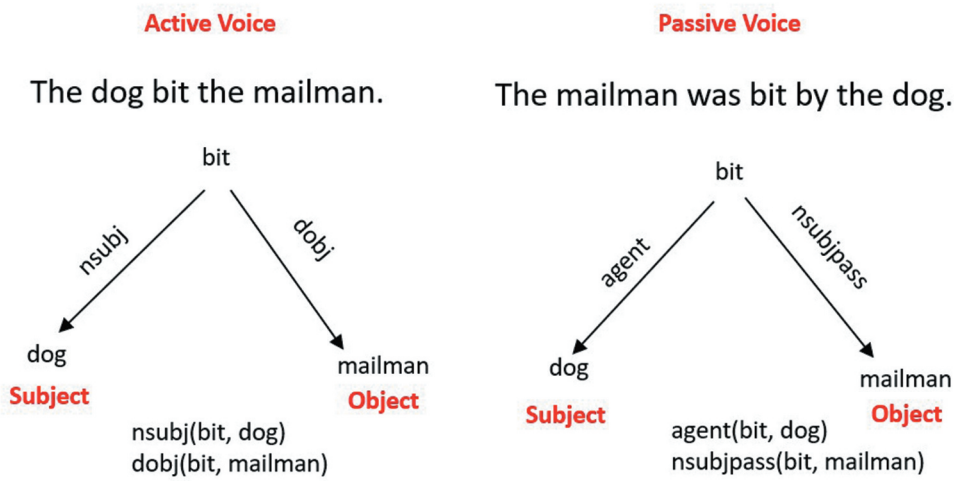

Figure 2. Subject and object dependencies from CoreNLP.

and other contexts in each sentence of the story. And the frequency counts contribute to each word's (or phrase's) weight in the story, compared to others in the same set. For example, the weight of a subject is based on its own frequency in the story, but also in the numbers of actions it takes and how many objects it affects. Similar metrics are used for weights of other types (objects, places, times, other contexts). Story analyzer considers each action to be an individual event, so all actions have the same weight, because each happened only once.

JavaScript code for the visualization module relies heavily on Data Driven Documents (d3),[29] a sophisticated API based on scalable vector graphics. SVG[33] is an XML-based framework that is DOM-compliant and therefore well-suited for browser-based visualizations. Although there are simpler tools and APIs for charting (e.g. Tableau, Google Charts), d3 has the advantage of allowing developers to create customized visualizations with full control over each SVG element in the chart. This fine-toothed control makes D3 an ideal tool for the visualization tasks required in Story Analyzer. Story Analyzer currently makes use of five publicly available D3 charts that are available: word clouds, chords, timelines, force graph, and s. In addition to the D3 visualizations, Story Analyzer also displays Google bar charts and a Google map. These are shown and described in the following section.

Accuracy limitations and workarounds

Note that the logic for identifying the subjects, objects, and other elements related to an action, and even the logic for finding the actions themselves, relies on the accuracy of the underlying NLP engine. Thus, the F1 scores of CoreNLP's annotators carry forward to influence the accuracy of the results shown in Story Analyzer. None are perfect, and as mentioned earlier, coreference resolution is especially prone to error.

An F1 score is a measure of accuracy in machine learning classification models and is used for evaluating the NLP models. F1 is an average of precision and recall. Precision is the proportion of positive predictions that are actually correct. Recall is the proportion of actual positives that were predicted. Oftentimes the goals of precision and recall conflict with one another. Law enforcement is a classic example of this tension. If you cast your net too wide and assume guilt, you'll catch more criminals, but you'll also catch many innocent people (low precision). On the other hand, if you lean too far into the presumption of innocence, you'll be less likely to unfairly prosecute innocent people at the cost of letting some criminals go free (low recall). This dichotomy is common to all supervised machine learning problems, including NLP models.

Many of the relevant F1 tests for NLP evaluation are done with respect to CoNLL and MUC data,[34] and CoreNLP's annotators score competitively with other leading models. Table 1 shows the F1 scores for the key CoreNLP annotators used by Story Analyzer.

Story Analyzer's information extraction algorithm attempts to mitigate for some common NLP errors, based on experiences with different documents over its development process. But for the most part, corrections to NLP results must be done manually. This can be done using the editing features

Table 1. Accuracy measures of CoreNLP annotators.

CoreNLP Annotator	F1 Score	Test Information Source
POS Tagging	97	https://nlp.stanford.edu/software/pos-tagger-faq.html
Dependency Parsing	81	https://nlp.stanford.edu/software/stanford-dependencies.shtml
Named Entity Recognition	81	https://nlp.stanford.edu/software/crf-faq.shtml
Coreference Resolution – neural network	60	https://stanfordnlp.github.io/CoreNLP/coref.html

available in **Story Analyzer Lite**, a desktop application that enables a user, who plays the role of "NLP editor", to identify and correct errors, as shown below.

The top screen in Figure 3 shows the coreference chains and mentions. Corrections can be made, including adjusting phrase lengths, reassigning mentions to other chains, creating new chains, and adding or removing mentions. This can have a significant impact, enhancing the quality of resulting dashboards. Similarly, as shown in the lower screen, editors can adjust parts of speech or named entity types as needed, either for individual sentences or globally in the document. Corrections of incorrect dates, times, and locations are also possible using this tool.

Visualization technologies used by story analyzer

Story Analyzer's user interface rests largely on Data Driven Documents (D3), which is a Javascript API for working with Scalable Vector Graphics (SVG). SVG is an XML-based standard for representing DOM-compliant element that can be embedded with HTML on a web page. D3 is the JavaScript API for working with SVG elements. Using SVG and D3 it is possible to create very powerful visualizations by combining and grouping a variety of basic interactive elements such as lines, rectangles, circles, polygons, and text elements into higher-level multi-element graphics. Using D3, the programmer has fine-tooth control over not just the chart as a whole, but all the underlying layers of elements that make up the chart.

The D3 visualizations used in Story Analyzer include a chord, a timeline, a force graph, and several word clouds. Story Analyzer also uses Google Maps technology. These will be described in the following sections.

Chord visualizations are circular structures where the periphery (the outline) of the circle is broken into segments. These segments depict specific entities that are being represented in the visualization. In Story Analyzer, the segments represent people and groups, and the size of the segment indicates that person of group's impact in the story. The periphery segments are connected by "chords" which are bands that connect pairs of segments in the visualization. Story Analyzer uses the chords to depict the interactions between pairs of people or groups in the story.

Force-directed graphs include nodes and links and depict networks of relationships. A complex network can have many nodes with varying numbers of links with other nodes, so the optimal placement of nodes in the visualization is a complex problem. The force-directed graph includes an algorithm

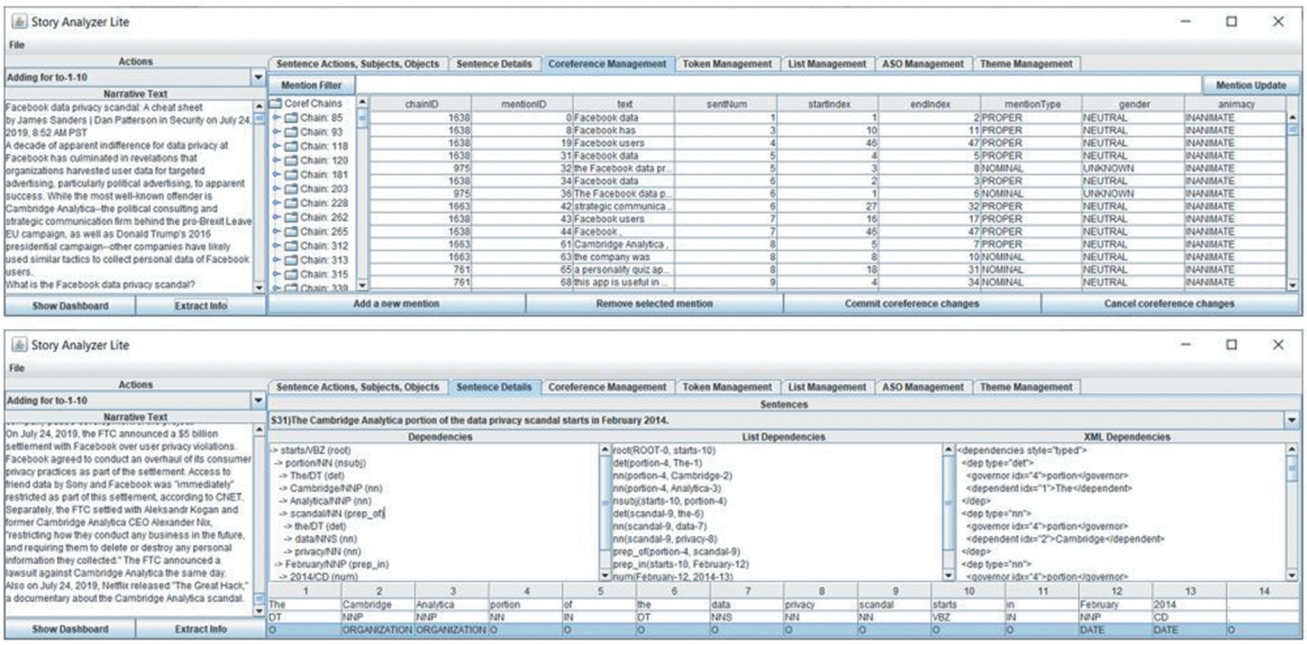

Figure 3. Story analyzer lite editing software.

based on "gravity" that attempts to find the optimal arrangement of the nodes and links; this is a service provided by D3.

The word clouds and timeline included in Story Analyzer are examples of third-party code samples that build on the base D3 library. In principle, it is possible to create a wide variety of custom visualizations through the use of D3 and the manipulation of basic SVG elements.

In the following sections, the use and capabilities of these visualizations will be described.

User experience with story analyzer dashboards

In sum, Story Analyzer identifies the following main types of story entities: actions, subjects, objects, people, groups, times, locations, and other contexts. Subjects and objects are often people and groups, but could also be locations or other nouns. Subjects perform the actions and objects are directly acted upon or interacted with indirectly within the dependency relations of a sentence.

A Story Analyzer dashboard consists of six sections, containing a total of sixteen interactive visualizations:

(1) Narrative and Highlighted Information
(2) People, Groups, Interactions, and Narrative Web
(3) Dates and Times
(4) Locations
(5) Subjects, Actions, and Objects
(6) Verbs, Nouns, and Contexts

The description and figures below pertain to a dashboard generated from a July 2019 TechRepublic article about the Facebook and Cambridge Analytica scandal (see https://www.techrepublic.com/article/facebook-data-privacy-scandal-a-cheat-sheet/).

Narrative/highlighted information and people/groups/interactions

The left panel of this section contains a high-level view of the Narrative as a whole. The right panel contains Highlighted Information based on a user's selection in the various visualizations. As you hover over a sentence within the narrative, you will see it automatically highlighted on the right. In fact, hovering the mouse over any of the dashboard's visualizations causes the relevant sentences to be highlighted in this section. Phrases are color-coded to depict either a person (blue), a group (red), a location (yellow), or a date or time (green).

Figure 4 shows a dashboard from the with both the Narrative and People/Groups sections open. In this figure, **Facebook-1-1** is selected from the list of groups, and the sentences having to do with Facebook are shown in the Highlighted Information section.

Every entity in the story, whether a person, group, place, or time, has two associated numbers attached to its label. The rightmost number is the sentence number where the entity is first found in the story (head coreference mention), and the number to its left is the token position of the main word in the phrase. Here you see that Facebook is first found in sentence number 1, at token number 1 in this sentence.

The lists of people and groups at the left of this section are paired with their impact within the narrative. This impact measures not only the number of mentions within the narrative, but also the number of actions and impacting subjects or impacted objects for the person or group. Higher impact actors are at the top. The circular **Interactions** visualization is a *chord* diagram. The bands along the circumference represent people (blue) or groups (red) in the story, and the connections (chords) between these bands represent the interactions between them. These relationships can occur between two people, two groups, or a person and a group. The final

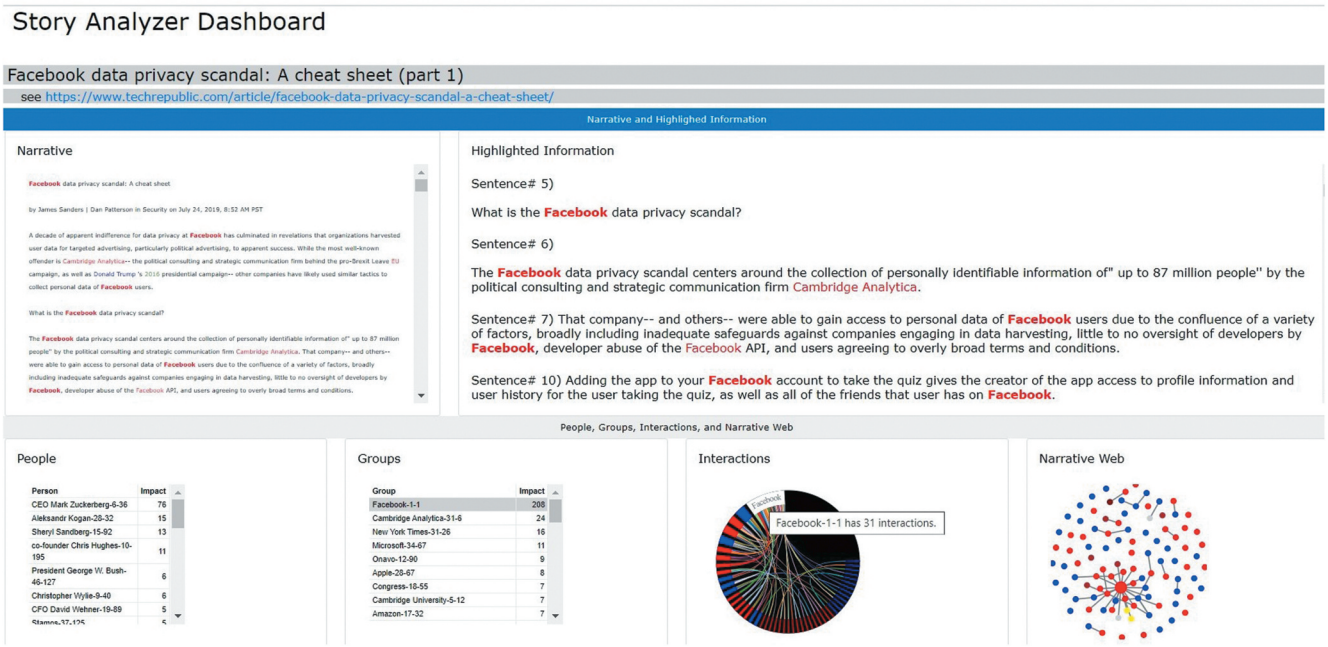

Figure 4. People, groups, interactions, and narrative web section of a story analyzer dashboard.

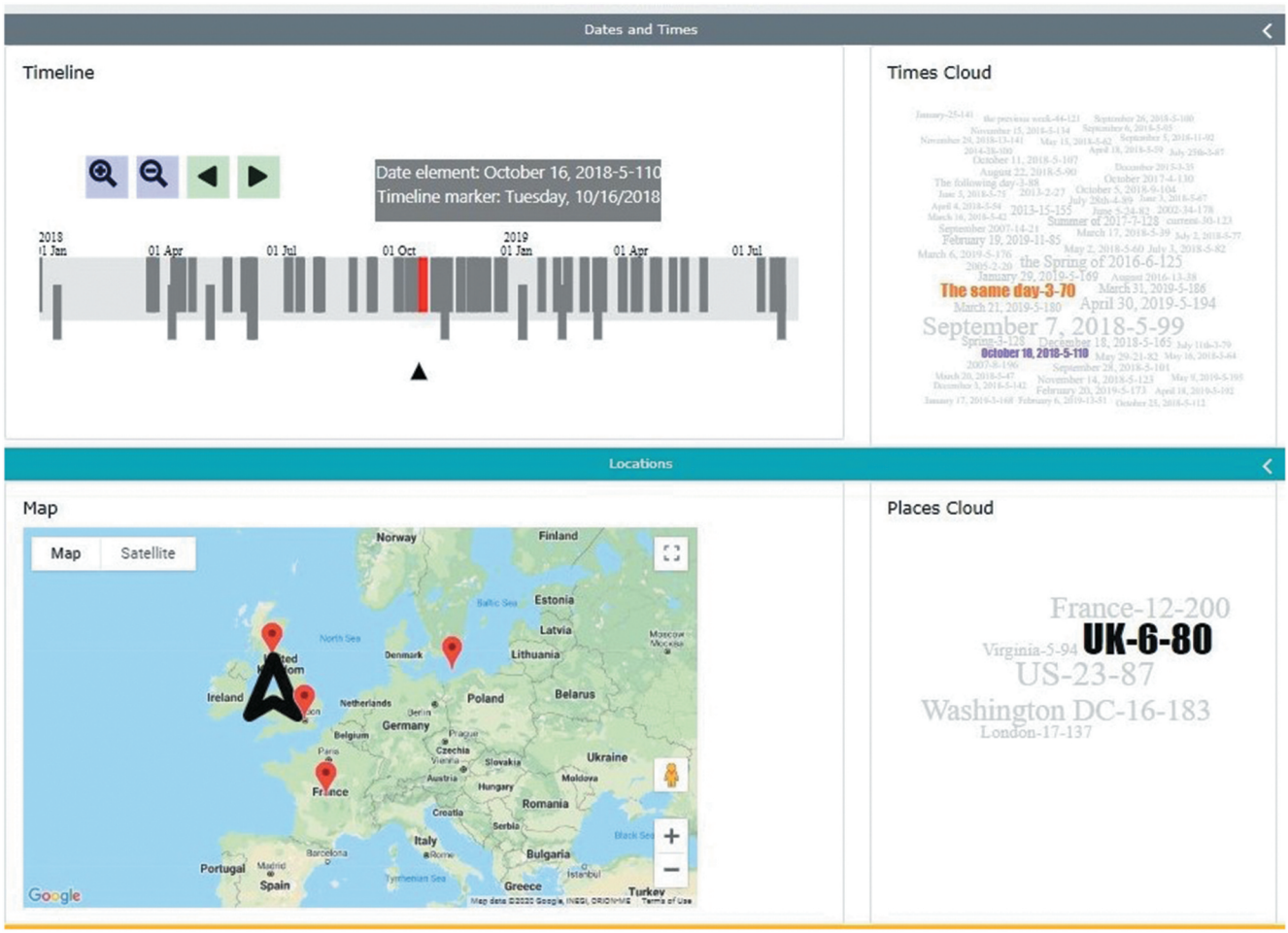

Figure 5. Dates/times and locations sections of a story analyzer dashboard.

visualization in this section is the **Narrative Web**. This is an example of a *force-directed graph*. Nodes in this web represent people, organizations, countries, and other entities, color coded for visual clarity. Hover over a node and a tooltip will appear containing the word and entity type. The links between nodes represent interactions between these entities. Generally, the bulk of the action will be clearly visible in a cluster of nodes in the center of the web.

Figure 4 illustrates the interactivity of the dashboard. Hovering the mouse over a name in the list will cause related elements in the other visualizations to be highlighted and will bring the relevant sentences into focus in the Highlighted Information section. In general, every visualization in a dashboard affects other visualizations.

Dates, times, and locations

Story Analyzer captures temporal information from CoreNLP using two annotators, the named entity recognizer and the temporal (SUDateTime) annotator. The Date and Times section of the dashboard includes a d3 timeline visualization and a word cloud to convey temporal information, as shown in Figure 5.

CoreNLP's SUTime annotator (https://stanfordnlp.github.io/CoreNLP/sutime.html) attempts to interpret a temporal phrase and generate a date string to it. Some date strings will be precise, like 2020–02-25. Others may be incomplete like 2016-XX-XX or convey a season, like 2015-FA. SUTime allows you to keep track of a "current" time in the narrative, and use this as a basis for interpreting phrases like "that afternoon" or "the previous week" or "a month later", which makes it possible to construct a linear timeline based on a variety of time-related phrases. The timeline visualization on the left is based on d3 code obtained from this site: https://codepen.io/chris-creditdesign/pen/yuFjr. The Times cloud is one of six word clouds found in Story Analyzer, all of which were adapted from code obtained at https://github.com/jasondavies/d3-cloud.

Note again the interactivity of the dashboard. Selecting a bar from the **Timeline** or a temporal entity from the **Times Cloud** brings the relevant sentences into view and highlights other word clouds in the dashboard, described below.

CoreNLP's named entity recognition (NER) annotator identifies several types of locations, including names of cities, states or provinces, and countries. It also recognizes other names, such as "Empire State Building" or "White House" as locations. Story Analyzer captures location-related named entities and sends these phrases to Google's geocoding web service (https://developers.google.com/maps/documentation/geocoding/intro), which returns the latitude/longitude location of the location phrase. Based on this data, Locations section of the dashboard presents a Google map with markers at the designated locations. In addition, another word cloud is shown with the location entities of the story. Remember, all Story Analyzer entities include the token and sentence numbers in their labels.

As with other visualizations, hovering a mouse over an element causes relevant sentences to be highlighted. Clicking on any element in the dashboard toggles between freezing and unfreezing the screen.

Subjects, actions, and objects

This section includes three word clouds, containing subjects, actions, and objects respectively. These are dispayed in Figure 6. This section is another way to visualize the interactions that take place between subjects and objects in the narrative. The size of the subject or object in a word cloud depicts its impact in the story. For subjects, this correlates with how many actions the subject performs and the number of affected objects. For objects, the size relates to how many actions are performed upon the object by other subjects.

When the user hovers over a word or phrase in a word cloud, the related phrases in others are highlighted. Colors connect actions to their associated objects.

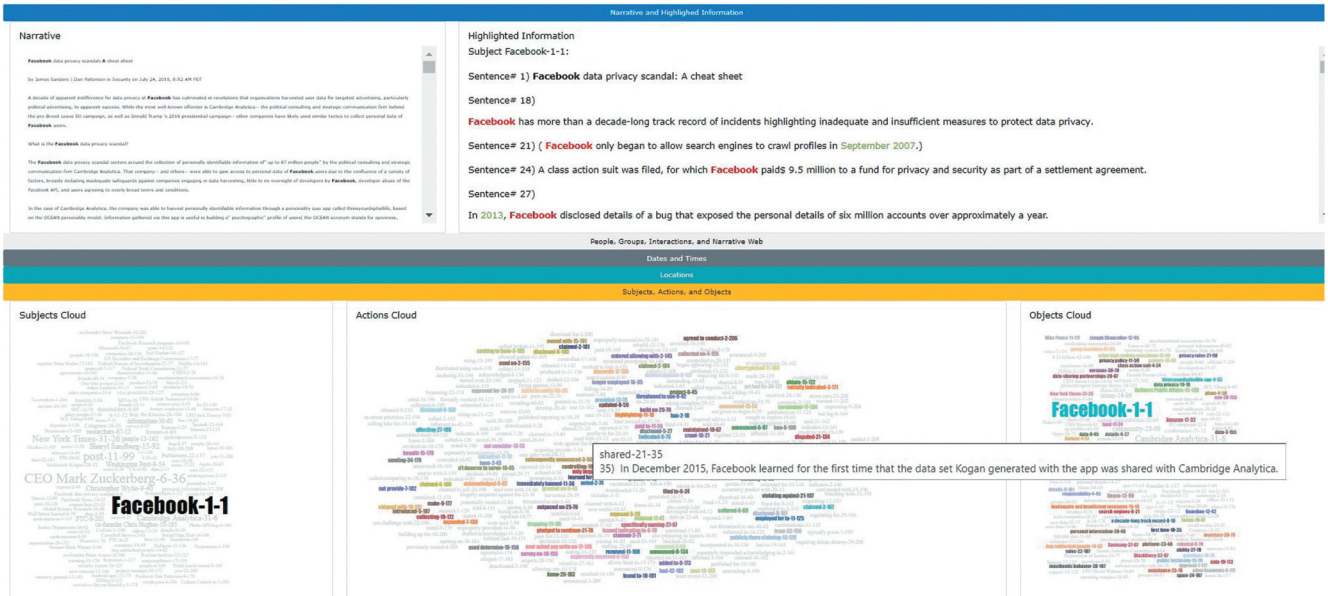

Figure 6. Subjects, actions, and objects section of a story analyzer dashboard.

Verbs, nouns, and contexts

The final section of the dashboard contains lists of the root forms (lemmas) of the verbs and non-name nouns in the document, along with their frequencies. This can help users see the most frequent terms (nouns and verbs) of interest. The common noun forms in this story include users, data and information, apps, privacy, and scandal. These common terms can give an idea of a story's theme, which is discussed below. In addition, this section includes another word cloud containing the contexts of actions in the story.

Incorporating themes into a story analysis

The above figures and description address the core elements of Story Analyzer, specifically the actions that occur, the people or groups that perform the actions (subjects), those that are affected (objects), and the contextual background including time, place and other contexts. But sometimes the reader of a story may have a particular interest when reading, a set of ideas or concepts through which to view the story. Or, as described above, you can identify common concepts of the story based on common terms. In other words, oftentimes a narrative has an embedded **theme**, or perhaps a reader has a particular thematic interest when reading the story.

Story Analyzer includes a feature that allows applying one or more themes to a story.

A theme is a cluster of concepts of interest, each with associated synonyms or related terms. A Story Analyzer dashboard can include a word cloud for a theme. Dashboard sections can be customized accordingly, as shown in Figure 7. Here we see that the narrative web is replaced with a theme cloud.

This dashboard was generated based on an online ethics discussion with posts from students in a senior-level computer information systems course. Students were asked to analyze the above mentioned TechRepublic article based on? the "Eight Key Questions", an initiative at our university to foster students' ethical reasoning skills (see https://www.jmu.edu/ethicalreasoning/8-key-questions.shtml). These are eight "lenses" through which to analyze complicated ethical scenarios. The eight lenses are fairness, outcomes, rights, responsibilities, liberty, empathy, authority, and character.

The full text of the discussion was run through Story Analyzer. Then the 8 Key Questions theme was applied to the results, and the resulting dashboard visualizes this discussion. As shown in Figure 7, the concept of liberty includes other related concepts like freedom and consent. Similar related terms are associated with the other seven concepts.

Story analyzer dashboard creation process

The process of developing a dashboard using Story Analyzer involves the following steps:

(1) Clean and prepare text
(2) Run against **Story Analyzer** for NLP processing
(3) Repeat until accurate enough:
    a. Use **Story Analyzer Lite** to edit NLP results and correct NLP errors
    b. Run Information Extraction (producing A/S/O and other Story Analyzer data)
    c. Run Dashboard data generation (json data)
(4) Copy data to dashboard templates and json repository

This process was developed based on significant experience working with the above documents. In principle, the steps should not require technical expertise, but Story Analyzer editors need a basic understanding of the structure given by CoreNLP annotators. Currently, student research assistants are using this procedure for Story Analyzer dashboard creation.

The first step is cleaning the text data from a document. This can be a tedious task because the PDFs of these documents are often generated via optical character recognition (OCR) from hardcopy, which sometimes produces incorrect results. Once the text is cleaned, it is fed to **Story Analyzer**,

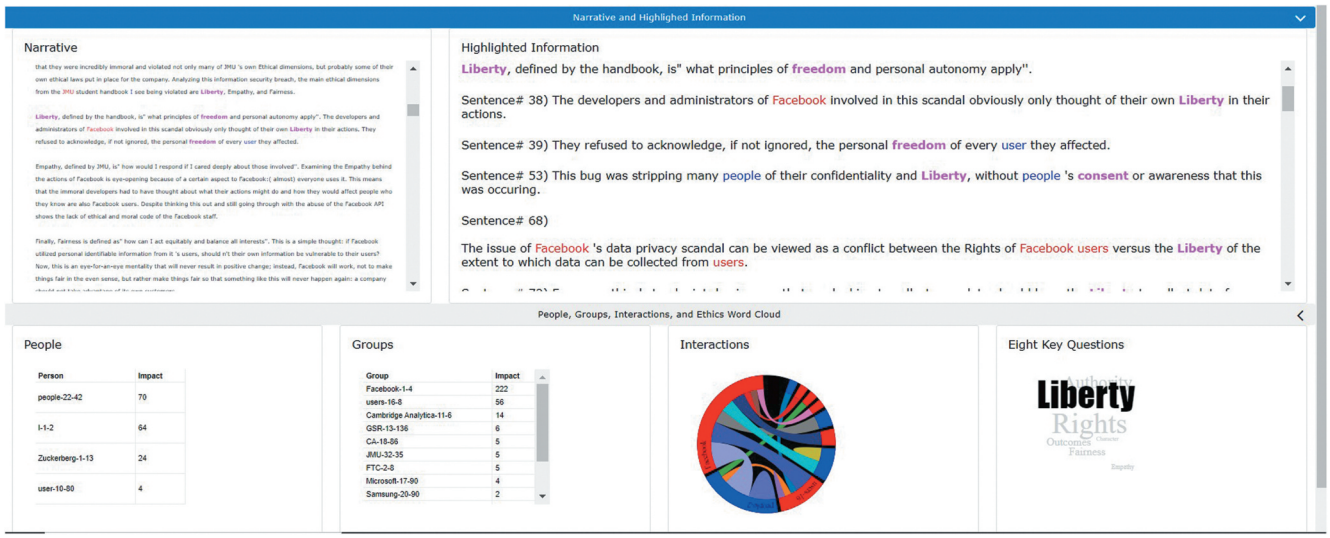

Figure 7. Dashboard with theme cloud.

which uses the CoreNLP engine to process results as described earlier. These results can the be stored into a story analyzer file.

As mentioned above, there are often errors in the CoreNLP results, so the third phase involves iteratively correcting and refining the NLP results. This is done using an editing tool called **Story Analyzer Lite**, with which a user can edit the results and test the information extraction and resulting dashboard. This can also be a time-consuming task, depending on how much correction needs to be done.

Finally, after reaching a suitable final result, the data can be pasted into a dashboard template and stored in a repository where it can be used for interactive display and analysis. Interested readers are encouraged to visit the Story Analyzer website (http://storyanalyzer.org/) for additional dashboards and information.

Conclusions and future work

This paper presents an application called Story Analyzer that combines Stanford's CoreNLP with D3 visualization functionality to assist users in quickly understanding the basic elements of a story, employing the maxim "a picture tells a thousand words." The application uses CoreNLP to split the text into sentences, tokenize the sentences, identify parts-of-speech and named entities, and find coreferences. The application also traverses each sentence's dependency network generated by CoreNLP's dependency parser. It attempts to depict the temporal sequencing of events, assisted by CoreNLP's temporal tagger SUTime. Using these tools, Story Analyzer attempts to extract story's subjects, actions, and objects, as well as each action's contextual factors such as place, time, and prepositional elements. D3 is then used to display these in word clouds. This application demonstrates the promise of NLP and visualization technologies to facilitate understanding of unstructured data in the form or story narratives.

A major limitation of Story Analyzer in its current state is that it contains no domain knowledge. For example, there is no knowledge that Obamacare and the ACA are one and the same thing, so these are treated as distinct entities. There is also no direct knowledge of what it means to be the president (definitions of concepts) or of the semantic relationships between concepts (categories and subcategories, parts of a whole, etc.) in the story. This kind of knowledge requires ontologies.

Future work with Story Analyzer will include the use of ontologies. The semantic structure underlying Princeton's WordNet[19] and/or ConceptNet[35] will be useful for adding depth and meaning to the features Story Analyzer can provide. Future work will also explore using sentiment analysis (another inexact NLP technology) to extract the emotional tones in a story. Other possible information extraction services include constituency parsing and natural logic semantics, both of which are services provided by CoreNLP.

Currently, Story Analyzer identifies Action/Subject/Object relationships in sentences, as well as possessions and copulas of an entity. It also includes some rudimentary temporal reasoning. Understanding a narrative should also involve recognizing the distinction between overt actions and internal thoughts or spoken utterances. Future work with this software will address these tasks.

There remain many limitations and challenges. The current state of NLP is less advanced and reliable than mining structured data. In addition, Story Analyzer is limited to a narrow branch of text understanding; it applies only to understanding a story. The application is not as well suited for extracting meaning from other types of text (e.g. ascertaining taxonomic hierarchies, understanding technical documents, legal reasoning, etc.).

Nevertheless, experiences developing this application indicate tremendous promise for applications using NLP services. As technologies advance and developers become more familiar with the tools available, opportunities for building useful software for extracting, and presenting meaning from text will continue to grow.

Acknowledgments

Thanks go to James Madison University's College of Business, who supported an educational leave in Fall 2015, during which this research began.

References

1. Schank RC. Computer understanding of natural language. Behav Res Methods Instrum. 1978;10(2):132–38. doi:10.3758/BF03205115.
2. Schank RC, Lebowitz M, Birnbaum M. An integrated understander. Am J Comput Ling. 1980;6:13–30.
3. Simmons LL, Conlon S, Mukhopadhyay S, Yang J. A computer aided content analysis of online reviews. J Comput Inf Syst. 2011;52:43–55.
4. Jurafsky D, Martin J Speech and language processing copyright © 2016. https://web.stanford.edu/~jurafsky/slp3/21.pdf.
5. Cowie J, Lehnert W. Information extraction. Commun ACM. 1996;39(1):80–91. doi:10.1145/234173.234209.
6. Grosz BJ, Spack Jones K, Webber BL, eds. Readings in natural language processing. Los Altos (CA): Morgan Kauffman; 1986.
7. Apache OpenNLP. 2020 [2020 May]. https://opennlp.apache.org.
8. The Berkeley NLP Group. 2020 [2020 May]. http://nlp.cs.berkeley.edu/software.shtml .
9. Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D The stanford coreNLP natural language processing toolkit. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations; 2014 [2014 June 22-27]; Baltimore, MD; 55–60.
10. Marcus MP, Marcinkiewicz MA, Santorini B. Building a large annotated corpus of english: the penn treebank. Computl Ling. 1993;19:313–30.
11. Conlon SJ, Abrahams AS, Simmons LL. Terrorism information extraction from online reports. J Comput Inf Syst. 2015;55(3):20–28. doi:10.1080/08874417.2015.11645768.
12. Conlon S, Hale J, Lukose S, Strong J. Information extraction agents for service-oriented architecture using web service systems: a framework. J Comput Inf Syst. 2008;48:74–83.
13. Balan S, Conlon S. Text analysis of green supply chain practices in healthcare. J Comput Inf Syst. 2016. doi:10.1080/08874417.2016.1180654.
14. Moretti G, Sprugnoli R, Menini S, Tonelli S. Extracting and visualizing content from large document collections to support humanities studies. Knowl-Based Syst. 2016;(111):100–12. doi:10.1016/j.knosys.2016.08.003.
15. Robeer M, Lucassen G, Van der Werf JMEM, Dalpiaz F, Brinkkempler S. Automated extraction of conceptual models from user stories via NLP. 2016 IEEE 24th International

Requirements Engineering Conference; 2016 [2016 Sept 12-16]; Beijing, China; 196–205.
16. Ceran B, Kedia N, Corman SR, Davulcu H Story detection using generalized concepts and relations. 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining; Aug 25-29; Paris, France; 942–49.
17. Zhu B, Wang Y, He C. Global social event extraction and analysis by processing online news. International Conference on Information System and Artificial Intelligence; 2016 [2016 June 24-26]; Hong Kong; pp73–76.
18. Chen H, Xu J, He B. Automated essay scoring by capturing relative writing quality. Comput J. 2014;57(9):1318–30.
19. Miller GA. Wordnet: a lexical database for english. Commun ACM. 1995;38(11):39–41. doi:10.1145/219717.219748.
20. Koch S, Marcus M, Muller A, Ertl T. VarifocalReader – in-depth visual analysis of large text documents. IEEE Trans Vis Comput Graph. 2014;20(12):1723–32. doi:10.1109/TVCG.2014.2346677.
21. Heimerl F, Lohmann S, Lange S, Ertl T Word cloud explorer: text analytics based on word clouds. 47th Hawaii International Conference on System Science; 2014 Jan 6-9; Waikoloa, Hawaii; pp. 1833–42.
22. Wang J, Zhao J, Guo S, North C, Ramakrishnan N. ReCloud: semantics-based word cloud visualization of user reviews. Graphics Interface Conference 2014; 2014 [2014 May 7-9]; Montreal, Quebec; pp. 151–58.
23. Lohmann S, Heimerl F, Bopp F, Burch M, Ertl T. ConcentriCloud: word cloud visualization for multiple text documents. 19th International Conference on Information Visualization; 2015 [2015 July 22-24]; Barcelona, Spain; pp. 114–20.
24. Collins C, Viegas FB, Wattenberg M. Parallel tag clouds to explore and analyze faceted text corpora. IEEE Symposium on Visual Analytics Science and Technology; 2009 [2009 Oct 12-13]; Atlantic City, NJ.
25. Mason DJ. Timeline visualization of semantic content. 8th ACM SigGraph Conference on Computer Graphics and Interactive Techniques; 2015 [2015 Nov 2-6]; Kobe, Japan; Poster Article #33.
26. Le DT, Vu NT, Blessing A. Towards a text analysis system for political debates. Proceedings of the 10th SIGHUM Workshop on Language Technology for Cultural Heritage, Social Science, and Humanities; 2016 [2016 Aug 11]; Berlin, Germany; pp. 134–39.
27. Xu P, Wu Y, Wei E, Peng T-Q, Liu S, Zhu JJH, Qu H. Visual analysis of topic competition on social media. IEEE Trans Vis Comput Graph. 2013;19(12):2012–21. doi:10.1109/TVCG.2013.221.
28. Heer J, Bostock M. Declarative language design for interactive visualization. IEEE Trans Vis Comput Graph. 2010;16 (6):1149–56. doi:10.1109/TVCG.2010.144.
29. Bostock M, Ogievetsky V, Heer J. D3 data-driven documents. IEEE Trans Vis Comput Graph. 2011;17(12):2301–09. doi:10.1109/TVCG.2011.185.
30. De Marneffe M-C, Manning CD. Stanford typed dependencies manual; 2008. http://nlp.stanford.edu/software/dependencies_manual.pdf.
31. Feng Y, Deng Q, Yu D. BLCUNLP: corpus pattern analysis for verbs based on dependency chain. 9th International Workshop of Semantic Evaluation; Proceedings of SemEval-2015; 2015 [2015 June 4-5]; Denver, CO; pp.325–28.
32. Chambers N, Cer D, Trond G, Hall D, Kiddon C, MacCartney B, de Marneffe M, Ramage D, Yeh E, Manning C. Learning alignments and leveraging natural logic. Proceedings of the Workshop on Textual Entailment and Paraphrasing; 2007 June; Prague; pp. 165–70.
33. Quint A. Scalable vector graphics. IEEE MultiMedia. 2003;10 (3):99–102. doi:10.1109/MMUL.2003.1218261.
34. CoNLL: the SigNLL conference on computational natural language learning; 2018 25. http://www.conll.org/July.
35. Speer R, Chin J, Havasi C. ConceptNet 5.5: an open multilingual graph of general knowledge. proceedings of AAAI 31; 2017 [2017 Feb 4-9]; San Francisco.