Story Analysis using Natural Language Processing and Interactive Visualizations

Mike Mitri
James Madison University

Abstract

This paper discusses the use of open source software libraries for extracting key story elements from a textual narrative and displaying them in an intuitive and interactive manner to users via a set of visualizations. The first library is Stanford’s CoreNLP, a Java API that provides natural language processing (NLP) services such as sentence recognition, tokenizing, parts-of-speech identification, dependency parsing, named entity recognition, coreference resolution, and temporal tagging. The second is D3, a JavaScript API built on scalable vector graphics (SVG) which provides powerful data visualization capabilities. Google Charts and Google Maps are also used. The paper describes an application called Story Analyzer that was built using these APIs. Application 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 software libraries, and it presents workaround when applied to story analysis. CoreNLP’s results, given a specific story of current news events, are analyzed in detail.

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, organizations, or other entities (subjects) performing actions that can affect other people, organizations, 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 (subject, objects, actions, time, place, and other contexts), and moreover to represent the relationships between these elements for each event 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. 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 word clouds showing the interaction between subject, object, actions, places, times, and other contexts. This kind of visualization, leveraging on the NLP functionality introduced above, can help a user quickly capture the essence of a story. Other visualizations are also useful for drawing a picture of a story, as will be described.

NLP involves a blend of artificial intelligence, computer science, 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) or recognizing named entities. At a deeper level, NLP attempts to extract or infer the underlying meaning of text; this has historically been termed “natural language understanding” (Schank, 1978; Schank et al., 1980).

A key element of natural language understanding, as applied to the story analysis task described in this paper, is information extraction (IE). IE 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” (Simmons et al., 2011). IE involves several tasks relevant to story analysis, including named entity recognition (NER), relation extraction, event extraction, temporal expression, and template filling (Jurafsky & Martin, 2016). IE is related to information retrieval (IR), but differs in an important respect (Cowie & Lehnert 1996). Information retrieval is used to identify relevant documents, a task often associated with search engines like Google. By 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 much 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 (Charniak, 1972; Schank, 1978; Schank et al., 1980). 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 (retrieved 2018), software tools from the Berkeley Natural Language Processing group (retrieved 2018), and Stanford’s CoreNLP annotators (Manning et al., 2014). 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 (Marcus et al., 1993)
- 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 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 (Simmons et al., 2011) and terrorism incident reports (Conlon et al. 2015). CAINES evolved from Flexible Information extRaction SysTem (FIRST) which was applied to analyzing textual NASDAQ financial information (Conlon et al 2008). 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 (Balan and Conlon 2016).

Another application is Analysis of Language and Content in a Digital Environment (ALCIDE) (Moretti et al 2016). ALCIDE is a web-based platform using a variety of NLP information extraction techniques to assist humanities scholars analyze literary and history documents. Visual Narrator (Robeer et al., 2016) applies NLP to the task of software requirements elicitation, analyzing user stories to generate OWL-based ontologies. Ceren et al (2015) describe an NLP application for detecting stories and themes embedded in longer online postings of extremist groups. Zhu et al (2016) apply information extraction to online news analysis. Chen et al (2014) 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 collocation analysis, topic modeling, key-phrase 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 (Miller, 1995).
**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 (Moretti et al 2016) combines word clouds with maps, timelines, and network graphs for various visual depictions of literary and historical text analysis. Zhu’s (2016) news analysis system uses these visualizations as well. Koch et al (2014) describe VarifocalReader, a multilayer visualization that hierarchically displays chapters, then sections, then word clouds of text content. Word Cloud Explorer (Heimerl et al. 2014) combines a central interactive word cloud; selecting a word from the cloud brings up other panel views and filters. ReCloud (Wang et al., 2014) uses word clouds for analyzing online reviews.

One example of combining multiple word clouds is ConcentriCloud (Lohman et al 2015) 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) (Collins et al 2009). 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 (Mason, 2015; Le et al., 2016; Xu et al. 2013). Stories also depict relationships between people, organizations, or other entities. Chord visualizations can be useful for this.

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 use of Stanford’s CoreNLP to extract key elements and relationships of a narrative, and the D3 JavaScript API (Heer and Bostock, 2010; Bostock et al., 2011) 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 algorithm 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 involves 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 two words in the sentence, called dependency relations (de Marneffe and Manning, 2008). 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 (de Marneffe and Manning, 2008). 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 (Feng et al., 2015) focusing on the most relevant dependencies. Use of Semgrex (Chambers et al., 2007), 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, 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) (Bostock et al., 2011), a sophisticated API based on scalable vector graphics. SVG (Quint, 2003) is an XML-based language 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 three D3 charts that are available on the web, one for word clouds, one for chords, and one for timelines. 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.

Story Analyzer’s Behavior on a Sample Story

Consider the following story. The sentences are numbered for reference in the analysis that follows.

11Before summer of 2016, almost nobody expected Donald J. Trump to be our next president. 12Although he had announced his candidacy the previous summer, many people thought this was just a publicity stunt at the time. 13Most pundits expected Hillary Clinton to win the presidency. 14But over time, Trump appealed to a core base of die-hard supporters, and these supporters proved us wrong. 15Trump stimulated nationalist sentiment by demonizing Islam and rebuking undocumented immigrants. 16He called for a travel ban against immigrants from Muslim countries and wanted a border wall on the Mexican border. 17Although many Americans were repulsed by this tactic, others appreciated his "America first" rhetoric.

18During the Republican primaries, Trump beat Marco Rubio in Florida and was defeated by John Kasich in Ohio. 19Throughout the campaign in 2016, he repeatedly referred to Rubio as "Little Marco", and he castigated Ted Cruz as "Lyin' Ted". 20Contrary to conventional wisdom, Trump beat Kasich, Rubio, Cruz, and all the other presidential candidates at the Republican national convention in Cleveland Ohio in July. 21Also in July, Clinton defeated Bernie Sanders for the democratic nomination. 22Ultimately, Trump beat Hillary in the general election on Nov. 8, although Clinton beat Trump in Virginia and in Maryland. 23Overall, the popular vote favored Clinton, but the Electoral College chose Trump.

24After his inauguration on January 20, 2017, Trump selected several cabinet appointees. 25He also fought with the press (who he calls the "dishonest media", and he tweeted several times to the American people. 26On January 30, Trump fired Acting Attorney General Sally Yates after she would not order DOJ employees to enforce President Trump's travel ban due to doubts over its legality. 27On January 31, Trump nominated Neil Gorsuch to the Supreme Court.

28In February, Congress approved many of Trump's cabinet appointees. 29Also in February, Trump fired Mike Flynn because Flynn had misled vice president Mike Pence in January about his December conversations with Russian officials.

30On March 5, Trump tweeted that Barack Obama had wiretapped his offices at Trump Tower. 31White House officials spent much of the following few weeks attempting to clarify Trump's claim. 32He continued to fight allegations of collusion with the Russian government. 33On March 20, FBI director James Comey confirmed that the FBI was indeed investigating the Trump campaign’s Russian ties. 34Ironically, Comey had also derailed Hillary Clinton's campaign by reporting about her emails in October of 2016.

35In April of 2017, the Senate confirmed the Gorsuch nomination, which was a major victory for Trump. 36Although Trump failed to completely dismantle Obamacare in his first 100 days as he had promised, the House of Representatives narrowly voted to repeal the ACA on May 4. 37On May 9, Trump fired Comey, stating that this firing relieved unnecessary pressure on the president’s ability to engage and negotiate with Russia. 38On May 24th the Congressional Budget Office estimated that the House health care bill would cause 23 million Americans to lose their insurance. 39The bill passed but the Senate failed to repeal Obamacare, which is still alive if not kicking. 40In December of 2017, Congress passed a significant tax cut bill, which was a victory for the president.
On May 17, 2017, Robert Mueller was appointed as a special counsel by Deputy Attorney General Rod Rosenstein. Since then, Mueller has been investigating Trump for possible collusion with Russian power-brokers. After a full year, Mueller hasn’t completed his investigation yet, and he is under considerable pressure to do so. Mueller is a Republican and is well respected by Democrats and Republicans alike. Trump is a Republican too, but he believes Mueller is conducting a witch-hunt.

Back in 2006, Trump had a sexual affair with porn start Stormy Daniels. During the campaign in 2016, Trump’s personal lawyer Michael Cohen had paid Daniels to refrain from publicizing her affair with Trump. But in early 2018, Stormy began to tell her story, aided by attorney Michael Avenatti. Trump vehemently denies that the affair ever happened. But on July 24, Cohen released an audio recording that suggests otherwise. In this recording, Trump discusses a payoff for Karen McDougal. McDougal had also claimed having an affair with Trump in 2006. These dalliances from the past are haunting Trump in 2018.

Currently, Trump is working with Korean and Chinese leaders, attempting to denuclearize the Korean peninsula. Secretary of State Mike Pompeo actively worked for a summit meeting between Trump and Kim Jong Un. Pompeo’s efforts came to fruition at around 10AM on June 12th when Trump and Kim held an historic meeting. Trump had backed out of Obama’s nuclear agreement with Iran, insisting that this agreement was a bad deal. So, it will be interesting to see the terms he sets with North Korea. Meanwhile, ISIS is weaker than before but continues to attack Iraqis, Afghanis, and Europeans. Israelis are still fighting Palestinians. Saudi Arabia keeps bombing Yemen.

On June 1st, the Trump administration started imposing tariffs against the European Union, Canada, and Mexico. His actions include a 25% tariff on imports of steel and a 10% tariff on aluminum. The tariffs angered U.S. allies, who planned retaliatory tariffs on U.S. goods, and heightened chances of a trade war. The same month, the immigration debate heated up considerably. Attorney General Jeff Sessions ratcheted up the pressure by separating illegal immigrants from their children. Most people see this as cruel and unfair, but some others think it’s a necessary part of border control. Trump wants his beautiful wall; perhaps he sees the kids as a bargaining chip.

During the campaign Trump had claimed that NATO is obsolete. This July, he again bashed NATO, claiming that NATO countries are not paying their fair share for defense. NATO allies are also worrying about Trump’s friendly overtures to Russia.

The American people elected Donald Trump. He is now the president of the United States. Now “the Donald” is leading us, and we are anxiously watching to see where he takes us. Will peace come to Korea? Will Trump try to fire Mueller? Will Flynn, Cohen, or Paul Manafort eventually flip against Trump? Will we become less divided about immigration? Time will tell.

After reading this, you have a sense of the story. This includes the actions that take place, the people or entities who perform these actions (subjects) and the people or entities who are affected (objects). You also have a sense of when and where these actions took place, as well as other contextual factors. You know the main characters, as well as groups of people involved.

When Story Analyzer reads this story, it produces the display of Figure 3. Note that Story Analyzer takes approximately two minutes to produce the results, most of which involves the CoreNLP annotators processing the story and generating their results.

The full narrative is shown in the left of the page, and six D3 word clouds (Davies, 2018) are seen in the middle. Two additional D3 visualizations are shown; one is a chord visualization (Bostock, 2018), useful for showing binary relationships between entities. The other is a timeline visualization (Chris-credit Design, 2018), which helps a user to see a linear display of the key dates and date ranges of the story. There are also four Google bar chart visualizations (Google, 2018a). The top two show the people and groups of the story, respectively, along with their impact factors, which measure the total impact of the person or group, both as subject and as object. The bottom two display word counts for POS types and for NER types in the story. Finally, a Google map visualization (Google, 2018b) displays the locations discussed in the story.

Each word or phrase in a word cloud includes two numbers. The rightmost number is the sentence number where the entity is first found, and the number to the left of this is the token position of the main word in the phrase. For
example, here you see that Donald J. Trump is first found in sentence number 1, at token number 11 (note: CoreNLP includes commas as tokens).

Within a cloud, the font size of a word or phrase is based on the weighting described above. For example, Trump is the dominant subject and is also the dominant object in this story. In other words, Trump does more things that affect more objects than any other subject. Similarly, Trump is the recipient of more actions from more subjects than any other object. This is shown in figures 4 and 5, where Trump is selected from the subjects cloud and the objects cloud respectively.

![Image](image_url)

**Figure 3. Initial display of Story Analyzer visualizations**

When a word or phrase is selected from a cloud, the related words/phrases in other clouds are also highlighted. The colors of related words indicate specific relationships. For example, by looking at the dark blue items in the clouds of figure 4 you see that Trump (selected subject in black) beat (action) Rubio, Kasich, Cruz, and other candidates (objects) in Cleveland (place) in July 2016 (time) during the Republican national conventions (other context), and that this was despite conventional wisdom (other context). The arrows in figure 4 are not part of the screen image; these are inserted to highlight the related word-cloud elements. Similarly, we see in figure 5 green coloring that shows Hillary Clinton (subject) beating Trump (selected object) in Virginia and Maryland (places). Story Analyzer also includes an animation feature so that hovering a mouse over an action causes related elements in other clouds to move. This support for fine-grain control of individual SVG elements is an advantage of D3 over simpler visualization environments like Tableau or Google Charts.

Note also that selecting Trump from the subjects cloud highlights every sentence in which Trump is a subject of an action. Similarly, selecting Trump from the objects highlights sentences in which an action was performed affecting Trump.

The user can also select a word/phrase from other clouds and see related items. If we want to see what happened in February 2017, we can select this from the times cloud, as shown in figure 6. Here we see that Trump fired Mike Flynn, and that Congress approved cabinet appointees. Note also that the sentences in question, #18 and #19, do not explicitly include the year. Story Analyzer keeps track of the most recent mentioned date in the story line and uses this as a reference point for subsequent date and time mentions. At this point in the narrative, the most recent date
mention that includes a year is January 20, 2017 in sentence #14. So, in the current context, these February mentions are assumed to be 2017. CoreNLP’s SUTime temporal annotator also comes in handy for temporal reasoning in the story. Consider “the previous summer” in sentence #2. Given the context of 2016 from sentence #1, SUTime correctly returns 2015 as the year. Sentence #55’s “the same month” receives similar treatment. However, temporal reasoning in Story Analyzer is very rudimentary at this point.

Figure 4. Donald J. Trump selected from the Subjects cloud

Figure 5. Donald J. Trump selected from the Objects cloud
Figure 6. February of 2017 selected from the Times cloud

The user can also select a specific sentence from the narrative to see the key relationships for that sentence, as shown in figure 7. Here we see two main actions of a sentence. First, Trump fails to dismantle Obamacare during his first days. Second, on May 4, the House narrowly repeals the ACA.

Many subject/action/object triples are straightforward, involving the same verb as the governor of both an nsubj dependency and a dobj dependency, where the nsubj dependent is the subject and the dobj dependent is the object.
Figure 2 shows one of these simple and direct subject-object relationships. At other time, the relationship is indirect, and requires navigating through a graph of dependencies. This is illustrated in figure 8. The dependencies shown are sentence #44. In this sentence, Trump is doing two things: (a) working with Korean and Chinese officials, and (b) attempting to denuclearize the Korean peninsula. In both cases, the object is not directly related to the subject via a dobj dependency. Rather, a path through the dependency graph includes intervening dependencies, such as a clausal complement (xcomp), a preposition (prep_with) and another dependency (dep). Prepositions are useful for several purposes. In addition to facilitating indirect paths from subjects to objects in an ASO triad, they are also indispensable for identifying places, times, and other contexts for a given action in the story.

As this figure also shows, other dependencies can be useful for giving more detailed information about the subjects or objects. For example, adjectival modifiers (amod) are used here to give more descriptive labels in the objects cloud. In other sentences, the compound noun dependency (nn) comes in handy; for example, to combine a person’s first and last name together into a full name (e.g. Donald J. Trump, Hillary Clinton). As you can see from the figure, after the subject is identified, coreference resolution associates the word “Trump” with the previous entity “Donald J. Trump” from the story’s first sentence.

Although much of Story Analyzer’s reasoning involves identifying and visualizing the ASO relationships described above, there are other important story elements that can be found using CoreNLP. Two are possessions and copulas. Possession dependencies indicate ownership, and they are frequently associated with possessive pronouns such as “his”, “her”, “its”, “their”. Copula dependencies involve words like “is”, “are”, “was”, “were”, “am”; in other words, the “to be” term. A copula can indicate identity (“Cheryl is my wife”) or set membership (“Trump is a Republican”). A copula can also indicate a state of being, such as “I am happy”. Figure 9 includes two additional word clouds showing possessions and copulas. In this case Robert Mueller was selected from the subjects cloud, and the tooltip shows that he has an investigation (possession) and is a Republican (copula).
Word clouds form the main visualization basis of Story Analyzer, but other visualizations are also useful. The chord visualization shown on the right of figure 10 is integrated with the rest of the dashboard. Along the circumference of the chord visualization are entities (in this case people) of the story. Each chord is a band connecting one entity with another, so the chord visualization is used for depicting binary relationships among a set of entities. Hovering the mouse over a chord displays a tool tip with the actions between these entities, and clicking the chord causes the appropriate word cloud items and sentences to be highlighted, as shown in figure. Similar interactions occur with the timeline, barcharts, and map.

Figure 10. Using the chord visualization

Limitations, Workarounds, and Future Work

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 binary 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 assumption of innocence, you’ll be less likely to unfairly prosecute innocent people at the cost of letting 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 (CoNLL, 2018), 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.

The coreference output from CoreNLP’s neural network coreference annotator for the above story is shown in the Appendix. This output shows chains of two or more mentions. A mention of a coreference chain contains a phrase consisting of one or more consecutive tokens from a sentence in the document. The first mention of a chain is the head mention, and subsequent mentions come from later sentences in the text. Each mention also includes other
properties, including animacy (animate, inanimate, unknown), gender (male, female, neutral, unknown), and mention type (nominal, pronominal, proper).

<table>
<thead>
<tr>
<th>CoreNLP Annotator</th>
<th>F1 Score</th>
<th>Test Information Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS Tagging</td>
<td>97</td>
<td><a href="https://nlp.stanford.edu/software/pos-tagger-faq.html">https://nlp.stanford.edu/software/pos-tagger-faq.html</a></td>
</tr>
<tr>
<td>Coreference Resolution – neural network</td>
<td>60</td>
<td><a href="https://stanfordnlp.github.io/CoreNLP/coref.html">https://stanfordnlp.github.io/CoreNLP/coref.html</a></td>
</tr>
</tbody>
</table>

Table 1. Accuracy measures of CoreNLP annotators.

If you carefully read the story and compare this to the coreference resolution output, you can see many instances where later mentions are correctly associated with a previous entity. For example, the output accurately links last name mentions with earlier full name mentions (e.g. Trump, Clinton, Kasich, Rubio, Comey, Flynn, and Mueller), and also correctly connects first names (Donald, Hillary). It identifies many useful links from pronouns to the correct entity reference. Sometime these correct coreference chains can seem downright intuitive. Consider chain #354 from the appendix. Identifying “we” and “us” to “the American people” works well in this story, as you can see in sentences #62 and #64.

However, you can also detect several precision errors (false positives) and recall errors (false negatives) in this set of coreference chains. For example, the coreference annotator incorrectly conflates the House of Representatives with the White House, a false positive (Appendix, chain #177).

In previous versions of the above story, the coreference resolution would fail to associate the first name “Hillary” in sentence #12 to Hillary Clinton (a false negative). Although the sentence in the older version is identical to the same sentence in the previous version current version, in the current story version coreference resolution succeeds (see chain 150 in the Appendix). The behavior of coreference resolution can be very quirky sometimes.

The coreference annotator incorrectly associates the “she” from sentence #16 with Donald Trump, when this pronoun should refer to Sally Yates (Appendix, chain #361). This is another false positive of matching “she” with Trump, and also indicates a false negative by failing to associate “she” with Yates. Interestingly, work by Rudinger et al. (2018) suggests gender bias in current coreference models. A similar mistake occurs with “his” in sentence 19; this makes it look like the vice president talked with Russian officials when Michael Flynn was the person who talked with them (Appendix, chain #122).

Consider again sentence #19. In the absence of domain knowledge (i.e. knowledge of current events and the actual characters of the story), it is difficult to know for sure which person had the conversation. Pronoun resolution is tricky because often the pronoun could refer to more than one entity with the sentence grammar still being correct. In this case, CoreNLP’s coreference resolution fails the Winograd Schema Challenge (Levesque, 2017). The WSC, named after famed AI researcher Terry Winograd, is a test for intelligence that was proposed as an alternative to the Turing test. Among other things, the WSC tests common-sense reasoning in pronoun resolution, such as the problem we see in sentence #19. Such common-sense reasoning is still a major stumbling block in NLP. Machine learning helps the coreference model learn patterns, but this learning is often wrong (hence the low F1 score). Associating the word “his” with the closest previous entity often makes sense, and would work in many grammatically similar statements, but in this case it is incorrect.

Another coreference error is seen regarding the month of July (Appendix, chain #335). The chain starts with July of 2016 in sentence #10, and the second mention correctly refers to the first. The third and fourth July mentions, however, are about July 2018, and should not be included in this chain.
Even if the coreference is valid, this does not mean you always want to substitute the coreference head for the mention. Possessive pronouns in coreference mentions can also cause problems. For example, in sentence #2, consider the phrase “he announced his candidacy”. In this case, the personal pronoun “he” is correctly associated with Trump so, using coreference resolution ensures the correct subject of the action “announced”. But, that same coreference chain also includes the phrase “his candidacy” as a mention. The word “his” correctly matches to Trump, and “candidacy” matches the direct object of the verb “announced”. But if we use the coreference head, then Trump himself, and not his candidacy, will become the object of the announcement. A similar problem occurs in a statement like “Mueller hasn’t completed his investigation”. So, in the case of possessive pronouns of objects in coreference mentions, Story Analyzer does not use the coreference head to determine the object of an action, but instead marks the object as a possession of the coreference head’s entity.

Not all errors are due to coreference resolution. For example, consider sentence #10. The tokens Trump, Kasich, Rubio, and Cruz are all correctly identified as named entities, specifically persons. But the dependency parser considers Kasich and Cruz to be a compound noun: nn(Cruz-12, Kasich-8). This implies a person whose first name is Kasich and whose last name is Cruz. In general, when a list of comma-separated named entity tokens is in a sentence, CoreNLP’s dependency parser tends to assign an nn dependency between two of them. But if the comma-separated list contains nouns that are not named entities, the mistake doesn’t occur, as in this sentence: “Cheryl eats fish, eggs, chicken, and other non-beef meats.”

Other difficulties arise when the POS tagger misidentifies ambiguous parts of speech. For sentence #66, the POS tagger correctly considers the word “fire” to be a verb. But if the question were instead “Will Trump fire Mueller?” it would incorrectly be tagged as a noun. This sentence also produces a NER error. “Will” is considered by the NER annotator to be the name of a person, although the POS tagger correctly interprets it as a modal verb (MD) and not as a proper noun (NNP).

These cases illustrate a central issue for software applications using NLP or other AI tools that often produce erroneous results. The correctness of the application is largely determined by the correctness of the underlying API. When working with an NLP service who’s F1 score may be in the 80s or even 60s, it is inevitable that the visualized results will sometimes give misleading information to the user.

To mitigate some of these errors, Story Analyzer includes algorithmic workarounds. One trick is to compare the gender of a pronoun against the gender of its coreference head mention. If these don’t match (as in the case of Sally Yates), then the coreference is probably not legitimate. In this case, the head mention is not used, and another form of pronoun resolution is required (Kruengkrai et al, 2014). When this gender mismatch occurs, Story Analyzer searches backward through the text from the pronoun in question, finding the most recent phrase recognized by CoreNLP’s NER annotator as a person, and then compares the name against a list known female or male names. The closest name prior to the pronoun which matches its gender is then used. This is how Story Analyzer correctly associates “she” with Sally Yates in the above story. Of course, this simplistic approach fails the WSC test, similarly to CoreNLP’s error with sentence #19.

To mitigate for significant mismatches of named entity mentions in a coreference chain (e.g. House of Representatives vs. White House), Story Analyzer evaluates the quality of the match. For example, a named entity should match in type the head mention’s named entity (e.g. person vs. organization). Also, the names can’t be too dissimilar from one another. In the case of House of Representatives (matched mention) vs. White House (head mention), both have the word “House” in them, but “White” is in the head only, and “Representatives” is in the matched mention only. When there is too much difference between the matched mention and its associated head mention, the coreference head recommended by CoreNLP is rejected.

To mitigate for the false negative of unmatched first names, Story Analyzer takes advantage of the incremental buildup of the story elements. If a subject or an object has already been identified with full name, and the first name doesn’t find a coreference head, Story Analyzer uses the matching subject or object as the matched entity. This only works if NER recognizes the first name as a name. In the above story, “Stormy Daniels” is recognized as a name, but “Stormy” as the first name alone is not. In this case, the NER annotator committed a false negative, failing to recognize Stormy as a named entity.

Another trick applies to the incorrect compound noun assumptions for comma-delimited named entity tokens (e.g. the Kasich, Rubio, Cruz example), as well as the “Trump and Sessions” error. When Story Analyzer encounters a comma-separated list of person or organization named entity tokens, it simply ignores the nn dependency and thus
does not create a compound noun for the subject or object; instead it treats the separate tokens as indicating separate entities. The same is done with “and” and “or” separators.

This would not work for dates, though (e.g. “July 4, 2018” has a comma). The same is true of locations (e.g. “Cleveland, Ohio”). The workaround may also cause errors for persons (e.g. “Donald Trump, Jr.”) and organizations (e.g. “IBM, Inc.”). So, correcting for one error can and often does introduce another.

The above analysis is based on CoreNLP version 3.9.1. CoreNLP is a work in progress, as is Story Analyzer. As NLP models become more accurate, their use in an application will become more reliable. But even in their current state, they provide very valuable information that can be expressed visually and help users come to a quicker understanding of text documents. Open source visualization APIs like D3 and Google provide opportunities for creating rich and intuitive interfaces to the information provided by NLP systems.

Story Analyzer has been informally tested on several stories found on the web throughout its development. Results are similar to those described here. Sometimes the coreference errors produce amusing effects. For example, in a Washington Post article from February 2017 (Entous et al., 2017), Sean Spicer is quoted as saying “The president is evaluating the situation. He’s speaking to Vice President Pence relative to the conversation the vice president had with Gen. Flynn and also speaking to various other people about what he considers the single most important subject there is: Our national security.” Elsewhere in the article is this sentence: “Yates, Clapper and Brennan argued for briefing the incoming administration, so the new president could decide how to deal with the matter.” In both cases, coreference resolution associates “president” with Vladimir Putin, not with Donald Trump. Although the phrase “President Trump” occurs early in the story, the phrase “Russian president Vladimir Putin” also appears later. This error obviously fails the WSC test for common sense reasoning. But it makes you think, doesn’t it?

Figure 11. Word clouds from Washington Post article, with Putin selected as a subject.

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 use of ontologies. The semantic structure underlying Princeton’s WordNet (Miller, 1995; Princeton University, 2010) 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 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.

REFERENCES


APPENDIX: CoreNLP Coreference Resolution Output

COREFERENCE CHAINS FROM NEURAL NET (only chains of two or more mentions are included)

Chain: 21
  die-hard supporters , => (Sentence 4 Tokens 12-14) NOMINAL UNKNOWN ANIMATE
  these supporters proved => (Sentence 4 Tokens 16-18) NOMINAL UNKNOWN ANIMATE

Chain: 50
  2016 , => (Sentence 1 Tokens 4-5) PROPER UNKNOWN INANIMATE
  2016 , => (Sentence 9 Tokens 5-6) PROPER UNKNOWN INANIMATE

Chain: 56
  John Kasich in Ohio . => (Sentence 8 Tokens 16-20) PROPER MALE ANIMATE
  Kasich , => (Sentence 10 Tokens 8-9) PROPER MALE ANIMATE

Chain: 57
  Marco Rubio in Florida and => (Sentence 8 Tokens 8-12) PROPER MALE ANIMATE
  Rubio as => (Sentence 9 Tokens 11-12) PROPER MALE ANIMATE
  Little Marco " => (Sentence 9 Tokens 14-16) PROPER MALE UNKNOWN
  Rubio , => (Sentence 10 Tokens 10-11) PROPER MALE ANIMATE

Chain: 58
  Ted Cruz as => (Sentence 9 Tokens 21-23) PROPER MALE ANIMATE
  Cruz , => (Sentence 10 Tokens 12-13) PROPER MALE ANIMATE

Chain: 105
  DOJ employees => (Sentence 16 Tokens 17-18) PROPER NEUTRAL INANIMATE
  its legality => (Sentence 16 Tokens 30-31) PRONOMINAL UNKNOWN INANIMATE

Chain: 117
  Mike Flynn because => (Sentence 19 Tokens 7-9) PROPER MALE ANIMATE
  Flynn had => (Sentence 19 Tokens 10-11) PROPER MALE ANIMATE

Chain: 118
  February , => (Sentence 18 Tokens 2-3) PROPER UNKNOWN INANIMATE
  February , => (Sentence 19 Tokens 3-4) PROPER UNKNOWN INANIMATE

Chain: 122
  vice president Mike Pence in January about => (Sentence 19 Tokens 13-19) PROPER MALE ANIMATE
  his December => (Sentence 19 Tokens 20-21) PRONOMINAL MALE ANIMATE

Chain: 143
  FBI director => (Sentence 23 Tokens 5-6) PROPER NEUTRAL INANIMATE
  the FBI was => (Sentence 23 Tokens 11-13) PROPER NEUTRAL INANIMATE

Chain: 150
  Hillary Clinton to => (Sentence 3 Tokens 4-6) PROPER FEMALE ANIMATE
Clinton defeated => (Sentence 11 Tokens 5-6) PROPER MALE ANIMATE
Hillary in => (Sentence 12 Tokens 5-6) PROPER UNKNOWN ANIMATE
Clinton beat => (Sentence 12 Tokens 16-17) PROPER MALE ANIMATE
Clinton , => (Sentence 13 Tokens 7-8) PROPER MALE ANIMATE
Hillary Clinton’s campaign => (Sentence 24 Tokens 7-10) PROPER FEMALE ANIMATE
her emails => (Sentence 24 Tokens 14-15) PRONOMINAL FEMALE ANIMATE

Chain: 153
Neil Gorsuch to => (Sentence 17 Tokens 7-9) PROPER MALE ANIMATE
Gorsuch nomination => (Sentence 25 Tokens 10-11) PROPER UNKNOWN ANIMATE

Chain: 171
FBI director James Comey confirmed => (Sentence 23 Tokens 5-9) PROPER MALE ANIMATE
Comey had => (Sentence 24 Tokens 3-4) PROPER MALE ANIMATE
Comey , => (Sentence 27 Tokens 7-8) PROPER MALE ANIMATE

Chain: 177
White House officials => (Sentence 21 Tokens 1-3) PROPER NEUTRAL INANIMATE
the House of Representatives narrowly => (Sentence 26 Tokens 18-22) PROPER NEUTRAL INANIMATE
House health => (Sentence 28 Tokens 11-12) PROPER NEUTRAL INANIMATE

Chain: 184
23 million Americans to => (Sentence 28 Tokens 17-20) PROPER UNKNOWN ANIMATE
their insurance => (Sentence 28 Tokens 22-23) PRONOMINAL UNKNOWN ANIMATE

Chain: 186
the House health care bill would => (Sentence 28 Tokens 10-15) NOMINAL NEUTRAL INANIMATE
The bill passed => (Sentence 29 Tokens 1-3) NOMINAL NEUTRAL INANIMATE

Chain: 187
the Senate confirmed => (Sentence 25 Tokens 6-8) PROPER NEUTRAL INANIMATE
the Senate failed => (Sentence 29 Tokens 5-7) PROPER NEUTRAL INANIMATE

Chain: 189
Congress approved => (Sentence 18 Tokens 4-5) PROPER NEUTRAL INANIMATE
Congress passed => (Sentence 30 Tokens 6-7) PROPER NEUTRAL INANIMATE

Chain: 191
2017 the => (Sentence 25 Tokens 5-6) PROPER UNKNOWN INANIMATE
2017 , => (Sentence 30 Tokens 4-5) PROPER UNKNOWN INANIMATE

Chain: 226
porn start Stormy Daniels . => (Sentence 36 Tokens 11-15) PROPER MALE ANIMATE
Daniels to => (Sentence 37 Tokens 15-16) PROPER MALE ANIMATE
her affair => (Sentence 37 Tokens 20-21) PRONOMINAL FEMALE ANIMATE
her story => (Sentence 38 Tokens 10-11) PRONOMINAL FEMALE ANIMATE

her affair with => (Sentence 37 Tokens 20-22) NOMINAL UNKNOWN INANIMATE
the affair ever => (Sentence 39 Tokens 5-7) NOMINAL UNKNOWN INANIMATE

an audio recording that suggests otherwise. => (Sentence 40 Tokens 7-13) NOMINAL NEUTRAL INANIMATE
this recording, => (Sentence 41 Tokens 2-4) NOMINAL NEUTRAL INANIMATE

Karen McDougal. => (Sentence 41 Tokens 10-12) PROPER FEMALE ANIMATE
McDougal had => (Sentence 42 Tokens 1-2) PROPER UNKNOWN ANIMATE

2006, => (Sentence 36 Tokens 3-4) PROPER UNKNOWN INANIMATE
2006. => (Sentence 42 Tokens 11-12) PROPER UNKNOWN INANIMATE

Secretary of State Mike Pompeo actively => (Sentence 45 Tokens 1-6) PROPER MALE ANIMATE
Pompeo’s efforts => (Sentence 46 Tokens 1-3) PROPER MALE ANIMATE

Kim Jong Un. => (Sentence 45 Tokens 15-18) PROPER NEUTRAL ANIMATE
Kim held => (Sentence 46 Tokens 16-17) PROPER MALE ANIMATE

Trump and Kim Jong Un. => (Sentence 45 Tokens 13-18) LIST UNKNOWN ANIMATE
Trump and Kim held => (Sentence 46 Tokens 14-17) LIST UNKNOWN ANIMATE

Barack Obama had => (Sentence 20 Tokens 8-10) PROPER MALE ANIMATE
the president’s ability => (Sentence 27 Tokens 17-20) NOMINAL MALE ANIMATE
the president. => (Sentence 30 Tokens 19-21) NOMINAL MALE ANIMATE
Obama’s nuclear => (Sentence 47 Tokens 6-8) PROPER MALE ANIMATE

Obama’s nuclear agreement with => (Sentence 47 Tokens 6-10) NOMINAL NEUTRAL INANIMATE
this agreement was => (Sentence 47 Tokens 15-17) NOMINAL NEUTRAL INANIMATE

tariffs against => (Sentence 52 Tokens 10-11) NOMINAL UNKNOWN INANIMATE
The tariffs angered => (Sentence 54 Tokens 1-3) NOMINAL UNKNOWN INANIMATE
illegal immigrants from => (Sentence 56 Tokens 11-13) NOMINAL UNKNOWN ANIMATE
their children => (Sentence 56 Tokens 14-15) PRONOMINAL UNKNOWN ANIMATE

Chain: 321
this as => (Sentence 57 Tokens 4-5) NOMINAL NEUTRAL UNKNOWN
it’s => (Sentence 57 Tokens 14-15) PRONOMINAL NEUTRAL INANIMATE

Chain: 332
the Trump campaign’s Russian => (Sentence 23 Tokens 16-20) NOMINAL NEUTRAL INANIMATE
the campaign Trump => (Sentence 59 Tokens 2-4) NOMINAL NEUTRAL INANIMATE

Chain: 335
July. => (Sentence 10 Tokens 29-30) PROPER NEUTRAL INANIMATE
July. => (Sentence 11 Tokens 3-4) PROPER NEUTRAL INANIMATE
July 24th => (Sentence 40 Tokens 3-4) PROPER NEUTRAL INANIMATE
This July, => (Sentence 60 Tokens 1-3) PROPER NEUTRAL INANIMATE

Chain: 339
NATO countries are => (Sentence 60 Tokens 11-13) NOMINAL UNKNOWN INANIMATE
their fair => (Sentence 60 Tokens 16-17) PRONOMINAL UNKNOWN ANIMATE

Chain: 340
NATO is => (Sentence 59 Tokens 8-9) PROPER NEUTRAL INANIMATE
NATO, => (Sentence 60 Tokens 7-8) PROPER NEUTRAL INANIMATE
NATO countries => (Sentence 60 Tokens 11-12) PROPER NEUTRAL INANIMATE
NATO allies => (Sentence 61 Tokens 1-2) PROPER NEUTRAL INANIMATE

Chain: 342
the Russian government . => (Sentence 22 Tokens 9-12) NOMINAL NEUTRAL INANIMATE
Russia. => (Sentence 27 Tokens 26-27) PROPER NEUTRAL INANIMATE
Russia. => (Sentence 61 Tokens 12-13) PROPER NEUTRAL INANIMATE

Chain: 354
The American people elected => (Sentence 62 Tokens 1-4) NOMINAL UNKNOWN ANIMATE
us . => (Sentence 64 Tokens 1-2) PROPER MALE ANIMATE
we are => (Sentence 64 Tokens 11-12) PRONOMINAL UNKNOWN ANIMATE
us. => (Sentence 64 Tokens 20-21) PRONOMINAL UNKNOWN ANIMATE

Chain: 355
North Korea. => (Sentence 48 Tokens 14-16) PROPER NEUTRAL INANIMATE
Korea ? => (Sentence 65 Tokens 5-6) PROPER NEUTRAL INANIMATE

Chain: 357
Robert Mueller was => (Sentence 31 Tokens 7-9) PROPER MALE ANIMATE
Mueller has => (Sentence 32 Tokens 4-5) PROPER MALE ANIMATE
Mueller has => (Sentence 33 Tokens 6-7) PROPER MALE ANIMATE
his investigation => (Sentence 33 Tokens 10-11) PRONOMINAL MALE ANIMATE
he is => (Sentence 33 Tokens 15-16) PRONOMINAL MALE ANIMATE
Mueller is => (Sentence 34 Tokens 1-2) PROPER MALE ANIMATE
Mueller is => (Sentence 35 Tokens 10-11) PROPER MALE ANIMATE
Mueller ? => (Sentence 66 Tokens 6-7) PROPER MALE ANIMATE

Chain: 359
Michael Cohen had => (Sentence 37 Tokens 11-13) PROPER MALE ANIMATE
24th Cohen released => (Sentence 40 Tokens 4-6) PROPER MALE ANIMATE
Cohen , => (Sentence 67 Tokens 4-5) PROPER MALE ANIMATE

Chain: 361
Donald J. Trump to => (Sentence 1 Tokens 9-12) PROPER MALE ANIMATE
he had => (Sentence 2 Tokens 2-3) PRONOMINAL MALE ANIMATE
his candidacy => (Sentence 2 Tokens 5-6) PRONOMINAL MALE ANIMATE
Trump appealed => (Sentence 4 Tokens 5-6) PROPER MALE ANIMATE
Trump stimulated => (Sentence 5 Tokens 1-2) PROPER MALE ANIMATE
He called => (Sentence 6 Tokens 1-2) PRONOMINAL MALE ANIMATE
his `` => (Sentence 7 Tokens 12-13) PRONOMINAL MALE ANIMATE
Trump beat => (Sentence 8 Tokens 6-7) PROPER MALE ANIMATE
he repeatedly => (Sentence 9 Tokens 7-8) PRONOMINAL MALE ANIMATE
he castigated => (Sentence 9 Tokens 19-20) PRONOMINAL MALE ANIMATE
Trump beat => (Sentence 10 Tokens 6-7) PROPER MALE ANIMATE
Trump beat => (Sentence 12 Tokens 3-4) PROPER MALE ANIMATE
Trump in => (Sentence 12 Tokens 18-19) PROPER MALE ANIMATE
Trump . => (Sentence 13 Tokens 14-15) PROPER MALE ANIMATE
his inauguration => (Sentence 14 Tokens 2-3) PRONOMINAL MALE ANIMATE
Trump selected => (Sentence 14 Tokens 10-11) PROPER MALE ANIMATE
He also => (Sentence 15 Tokens 1-2) PRONOMINAL MALE ANIMATE
he calls => (Sentence 15 Tokens 9-10) PRONOMINAL MALE ANIMATE
Trump fired => (Sentence 16 Tokens 5-6) PROPER MALE ANIMATE
she would => (Sentence 16 Tokens 13-14) PRONOMINAL FEMALE ANIMATE
President Trump ‘s travel => (Sentence 16 Tokens 21-24) PROPER MALE ANIMATE
Trump nominated => (Sentence 17 Tokens 5-6) PROPER MALE ANIMATE
Trump ‘s cabinet => (Sentence 18 Tokens 8-10) PROPER MALE ANIMATE
Trump fired => (Sentence 19 Tokens 5-6) PROPER MALE ANIMATE
Trump tweeted => (Sentence 20 Tokens 5-6) PROPER MALE ANIMATE
his offices => (Sentence 20 Tokens 12-13) PRONOMINAL MALE ANIMATE
Trump ‘s claim => (Sentence 21 Tokens 14-16) PROPER MALE ANIMATE
He continued => (Sentence 22 Tokens 1-2) PRONOMINAL MALE ANIMATE
Trump campaign => (Sentence 23 Tokens 17-18) PROPER MALE ANIMATE
Trump . => (Sentence 25 Tokens 19-20) PROPER MALE ANIMATE
Trump failed => (Sentence 26 Tokens 2-3) PROPER MALE ANIMATE
his first => (Sentence 26 Tokens 9-10) PRONOMINAL MALE ANIMATE
he had => (Sentence 26 Tokens 14-15) PRONOMINAL MALE ANIMATE
Trump fired => (Sentence 27 Tokens 5-6) PROPER MALE ANIMATE
Trump is => (Sentence 35 Tokens 1-2) PROPER MALE ANIMATE
he believes => (Sentence 35 Tokens 8-9) PRONOMINAL MALE ANIMATE
Trump had => (Sentence 36 Tokens 5-6) PROPER MALE ANIMATE
Trump 's personal => (Sentence 37 Tokens 7-9) PROPER MALE ANIMATE
Trump . => (Sentence 37 Tokens 23-24) PROPER MALE ANIMATE
Trump vehemently => (Sentence 39 Tokens 1-2) PROPER MALE ANIMATE
Trump discusses => (Sentence 41 Tokens 5-6) PROPER MALE ANIMATE
Trump in => (Sentence 42 Tokens 9-10) PROPER MALE ANIMATE
Trump in 2018 . => (Sentence 43 Tokens 8-11) PROPER MALE ANIMATE
Trump is => (Sentence 44 Tokens 3-4) PROPER MALE ANIMATE
Trump and => (Sentence 45 Tokens 13-14) PROPER MALE ANIMATE
Trump and => (Sentence 46 Tokens 14-15) PROPER MALE ANIMATE
Trump had => (Sentence 47 Tokens 1-2) PROPER MALE ANIMATE
he sets => (Sentence 48 Tokens 11-12) PRONOMINAL MALE ANIMATE
the Trump administration started => (Sentence 52 Tokens 5-8) NOMINAL NEUTRAL INANIMATE
Trump administration => (Sentence 52 Tokens 6-7) PROPER MALE ANIMATE
His actions => (Sentence 53 Tokens 1-2) PRONOMINAL MALE ANIMATE
Trump wants => (Sentence 58 Tokens 1-2) PROPER MALE ANIMATE
his beautiful => (Sentence 58 Tokens 3-4) PRONOMINAL MALE ANIMATE
he sees => (Sentence 58 Tokens 8-9) PRONOMINAL MALE ANIMATE
Trump had => (Sentence 59 Tokens 4-5) PROPER MALE ANIMATE
he again => (Sentence 60 Tokens 4-5) PRONOMINAL MALE ANIMATE
Trump 's friendly => (Sentence 61 Tokens 7-9) PROPER MALE ANIMATE
Donald Trump . => (Sentence 62 Tokens 5-7) PROPER MALE ANIMATE
He is => (Sentence 63 Tokens 1-2) PRONOMINAL MALE ANIMATE
the Donald " => (Sentence 64 Tokens 3-5) PROPER MALE ANIMATE
he takes => (Sentence 64 Tokens 18-19) PRONOMINAL MALE ANIMATE
Will Trump try => (Sentence 66 Tokens 1-3) PROPER UNKNOWN ANIMATE
Trump ? => (Sentence 67 Tokens 12-13) PROPER MALE ANIMATE