Graphical Viewing of Relationships Extracted from Online Articles

Jeffrey Rego
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233 USA
jrego@mail.utexas.edu

Abstract

This paper discusses an approach to extracting and viewing relationships between named entities from news articles. It covers the preprocessing, parsing, extraction, filtering, and visualization of this information, starting from online news articles and ending with a visual graph of concepts they contain.

1 Introduction

With the advent of Web 2.0, content delivered in a machine-readable manner has become more popular in comparison to other delivery formats that target the end reader only. Services that assume that the data will be delivered exactly in the format it was published limit their audience and do not facilitate other uses (Miller, 2005). Machine-readable data and meta-data allows content providers to produce mash-ups of multiple data sets, such as the combination of news articles with maps of the locations they are about. This data can be used to combine several articles into one meta-view, or search a group of articles for a concept rather than a string of text. News articles can be visualized as hyperbolic graphs, with key concepts and entities as nodes, and relationships between them as the connections. This allows a user to explore the key concepts and how they relate to everything else in current events more effectively and interactively.

News aggregation could also be automated. There are currently many popular news aggregators, like the Drudge Report and Google News. They provide headlines, and sometimes short summaries of the articles, but they do not allow searching and grouping by arbitrary concepts. The user is limited to the grouping that the news aggregator chose when putting together the page. They also are not able to display multiple articles at a time, even when the articles are about the same topic.

A graph of concepts and relations is a more effective way to browse the aggregate content and easily manipulate the format it is displayed in (Mukherjea). Such graphs allow the data to be organized in abstract and easily manipulatable chunks, instead of large unstructured blocks of text. Thus, data from different sources could easily be matched together and displayed.

Machine-readable data also makes it possible to tailor the view to different devices. Today online news is not only accessible through a computer, but also through cell phones, handheld computers, and, maybe sometime in the near future, wearable computers and eBooks. Each device lends itself to different ways of displaying data. Cell phones have very low resolution screens, while handheld computers have the benefit of a stylus to manipulate the interface. If the core entities and relations can be identified in a new article and delivered in a format like eXtensible Markup Language (XML), simple formating code, like Cascading Style Sheets, can display the same article for any device.

Current technologies in computational linguistics make it possible to automatically analyze raw text to recover representations of the logical relationships that hold between entities. Using these techniques, a set of relevant entities and relationships can be extracted from existing bodies of text, and used for displaying in new formats.

2 Motivation

When researching a new topic, especially in current events, it can be helpful to read several articles on the same topic. For example, Google News returns more than a thousand articles related to Blackwater, covering different aspects of the same basic idea. My goal is to improve the visualization of the content of many articles at the same time via the relationships between the concepts presented. I extract these relationships from each article and display them all on a graph, allowing a reader to get an overview of the topic without having to read several complete articles.

There have been many projects that aimed to produce a summary from a body of text, often a news article. This topic is covered in several articles (Radev and McKeown, 1998b), (Dalianis, 2000), (Otterbacher et al., 2006). By picking out what appear to be the main topics and most important parts of the article, they can produce a short passage of text that provides an overview of the article. Many news aggregators today use text summaries to help the reader choose what news story they are interested in. Automated summarization offers more flexibility and would make this easier. For example, in some applications you would want longer summaries than in others, and certain types of information may be more important to a given person than other information. The ability to generate summaries on the fly would allow providers to tailor them to their users needs.

There are also projects aimed at combining information from news articles with other meta-data available on the web. There is a free online mashup of Google maps and local news stories, which will map each local news story based on the location it is from. It displays a map of the United States with a pushpin in each location a news article is from, allowing the user to choose articles from a geographic area of interest, or to easily see the distribution of current events. It is only able to map the location based on the tag for the article, and cannot get any more specific, or talk about multiple places. Extracting information from the article text itself would allow all locations represented in the article to be marked on a map.

Statements that are necessary to tie a news article together can cause visual clutter in a graph, which is better at displaying sparse data. This problem is related to text summarization, though the concepts that transfer from an article to a graph versus a text summary are different. Rather than seeing how a news agency came by all of the information that they have, and who said what, graphs are better at representing the key concepts and how they relate to each other.

Graph representation of the content of texts provides another view on such data. (Hensman and Dunnion, 2004) use VerbNet and WordNet to find a frame that matches each sentence, and then use the parsed and tagged sentence to extract meaning from the sentence. They filter out sentences which do not match any frames in VerbNet. The use of frames limits their coverage. This paper provides a process that does not rely on such a database, and thereby allows more information to be extracted from each article.

The kind of representation I produce is a graph of related concepts extracted by parsing the text and identifying named entities. Figure 1 shows an example produced from news article data relating to Blackwater. I use hyperbolic graphs in an interactive viewer to facilitate the display of a large amount of information centered around a certain concept (Lamping J., 1996). Concepts that are related to the central node radiate out from it, allowing more space and taking less of the focus as links get more numerous and less relevant to the central idea.

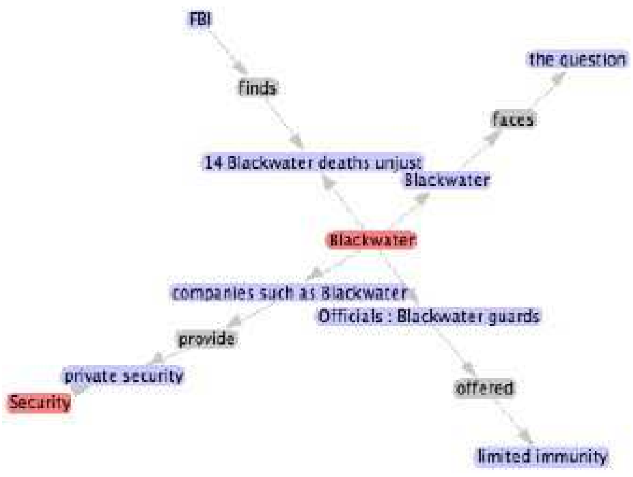

Figure 1: Example of graph of related concepts

3 Approach

This section covers in detail the steps that I go through from raw data to visualization. These steps are:

1. Obtain data
2. Preprocess data
3. Parse and extract relations
4. Filtering
5. Visualization

Finally, one shell script was written that runs each application in order with the appropriate inputs and outputs, allowing the final XML file to be produced in one step from a file containing the text of a number of articles.

3.1 Obtain data

A small set of ten news articles on the same broad topic were pulled from Google News. Google News provides good searching abilities, allowing easy retrieval of multiple articles on the same topic, as well as text summaries for each article. These features help in determining the performance of the application.

3.2 Preprocess data

The text from each article was added to one file, which was then processed by a custom application that converted newspaper-style text to a format that C&C (Clark and Curran, 2007) could read. It split all of the sentences in all articles up such that there was only one sentence per line, and punctuation that needed to be separated by spaces was separated. For example, C&C does not know how to handle a comma or period attached to a word, but an acronym like ”U.N.” should not have its periods detached. No effort was made to mark which article the text had come from, or where each article began and ended.

(1) For young veterans who loved military action but couldn’t afford to stay in , Blackwater offered big money and plenty of opportunities to order people around .

(2) As of last week , they said , some of the requested materials had still not been provided .

In order to speed up the process, as well as remove some of the human intervention, a small application was written to take the raw newspaper articles directly from Google News and convert them into a format that C&C can read.

Though C&C has been trained on news articles, the format it can read differs slightly from raw text. In particular, the parser operates on one sentence at a time, and expects to receive an input file containing sentences separated by line breaks. In order to break up an article, the preprocessor separates the text on sentence boundaries and output one sentence per line. It also adds spaces surrounding all punctuation, and removes punctuation from the middle of a string of numbers (25,000 becomes 25000). The most difficult step in this process is trying to determine which periods (.) are sentence boundaries and which are part of an abbreviation or acronym. A list of common abbreviations and acronyms, as well as matching the pattern of capitalized letters allows the preprocessor to successfully separate the sentences.

There are existing tools, such as OpenNLP, that also deal with the problem of sentence boundary detection. In the future, I would like to look into using an external solution rather than writing my own boundary detector. I believe that this would produce better results in this area.

3.3 Parse and extract relations

The output from reformatting all of the news articles is then passed through the C&C parser. C&C is a Combinatory Categorial Grammar parser capable of efficiently tagging and parsing sentences with parts of speech and grammatical relations. It has been trained on CCGbank (Hockenmaier and Steedman, 2007), which contains news articles, making it an appropriate choice for the domain we have chosen.

C&C parsed, supertagged, and identified named entities. This tags inline all of the named entities and their type (person, organization, time, etc.), the parts of speech of each word, and the syntax of each word. Figure 2 shows one sentence tagged with syntax rules for each word, and one tagged with named entity recognition. Both Dick Cheney and Donald

```
But|CC|S/S Blackwater|NNP|N has|VBZ|(S[dcl]\NP)/(S[pt]\NP)
said|VBN|(S[pt]\NP)/NP its|PRP$|NP[nb]/N contractors|NNS|N
fired|VBN|S[pss]\NP in|IN|((S\NP)\(S\NP))/NP
self-defense|NN|N .|.|.

Military|JJ|O contractors|NNS|O such|JJ|O as|IN|O
Halliburton|NNP|O and|CC|O Blackwater|NNP|O are|VBP|O
the|DT|O brainchild|NN|O of|IN|O Vice|NNP|O President|NNP|O
Dick|NNP|I-PERSON Cheney|NNP|I-PERSON and|CC|O
former|JJ|O Defense|NNP|I-ORGANIZATION Secretary|NNP|O
Donald|NNP|I-PERSON Rumsfeld|NNP|I-PERSON .|.|O
```

Figure 2: Example of syntax and named entity resolution

```
(det contractors_5 its_4)
(dobj in_7 self-defense_8)
(ncmod _ fired_6 in_7)
(ncsubj fired_6 contractors_5 obj)
(dobj said_3 contractors_5)
(aux _ said_3 has_2)
(ncsubj said_3 Blackwater_1 _)
```

Figure 3: Example of dependency parse

Rumsfeld were correctly identified as named entities of type I-PERSON.

Both the supertagged output and named entity recognition were used to determine which parts of each sentence were the actor, patient, and predicate, and how to relate them all to each other. In the first sentence in Figure 2, the actor, or entity that the relation refers to, is ”Blackwater”. The predicate is ”has said”, and the patient is ”its contractors fired in self-defense”.

The output from the supertagger was used to find the actor and patient for each action. The output was used to reconstruct the parse tree for each sentence. In building trees of the sentences, the grammatical relations are used (Carroll and Briscoe, 2001). An example of this type of output is shown in Figure 3. This information, combined with the syntax of each word, makes it possible to use the dependency parse tree that C&C found to find which part of the sentence is the actor and which the patient, as well as the verb phrase that is the predicate between them.

The combination of those grammatical relations and the parse tree is then used to find the verbs in the sentence, and their actors and patients. The link between the actor and patient is then stored in a list, and can be searched for given any part of the actor or the patient. For example, if *President Bush signed a bill*, looking for *Bush* would turn up the relation between *President Bush*, *signed*, and *a bill*.

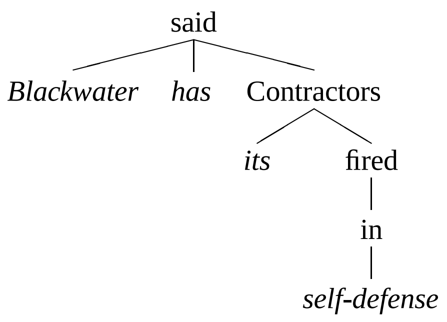

Figure 4: Dependency parse tree

After finding all actors, patients, predicates, and named entities, this application will output that data in a format ready to be read by the next steps in the process.

Figure 4 is a visualization of the parse tree created for the sentence, ”Blackwater has said its contractors fired in self-defense.” The C&C output that represents this tree is shown in Figure 3. Each time a grammar rule from C&C is processed, two parts of the sentence are linked, one as the parent and one as the child.

In order to improve the quality of the graph, I have been continuing to work on the actual output of the relations in the parsing step right after running C&C. If the tree is rebuilt incorrectly, each subsequent step in the process will not function properly, and many straightforward relations end up being filtered out because all the data is not there. Currently, the dependency parse trees sometimes have cycles in them. Tree nodes with two parents are not processed correctly, and often lead to the predicate containing some of the patient. I will need to either figure out how better to deal with these situations, or filter these sentences out. I will also need to filter out sentences whose patients end up too long to be reasonably displayed on a graph.

3.4 Filtering

A simple entity and relation parser is not enough to effectively create a graph of a given news article. Many of the relations returned will either be irrelevant when seen on a graph, or have incomplete information. Similarly to text summarization tasks, the relevant ideas from the article must be identified and included, and the extraneous information must be left out, otherwise interesting data will be lost in the noise caused by too much information.

After parsing the newspaper articles, a list of entities and relations are judged based on which ones were likely to be relevant when displayed on a graph. This helped to filter out sentences that were not relevant, as well as incorrect parses where all data was not present, and improve the quality of the output. Increasing the amount of data that goes in to the parser is easy, as it takes raw newspaper text with minimal human intervention and runs very quickly. Once the full text of an article has been added to a file for processing, all steps involved in creating a visualization can be completed in under a minute. However, unless it is possible to filter relations that should not be displayed, this extra data only results in a cluttered and confusing graph.

Parsing and extracting data produced a large number of entities and relationships between them. Many of these were not relevant to the topic of the article, and made it difficult to attempt to read through the results. In order to efficiently display the information in a news article it was be necessary to score each relation based on its relevance to the topic of the article and its relevance when displayed in the context of a graph, and only show the most relevant relationships.

Different features of both the original sentence and the parsed entity-relation output are used to determine a relevance score. This score reflects both the importance to the article, as well as whether or not it makes sense in a graph. The values for each feature will be set based on human review of a development set of relations. It would be possible to compute the article and graph relevance scores separately, but unnecessary in this context. Instead, a high score would indicate that a given relation is both relevant to the article and sensible to graph.

Looking at an individual sentence, it is possible to use multiple metrics to determine the likelihood that it would appear in a summary (Goldstein et al., 1999). Summary candidates are more likely to contain information central to the article, and are more likely to be interesting in a graph. The class of words found in the main entities within the sentence, which can be determined using WordNet, give a good indicator of importance.

The class of verbs used will also be considered when determining a sentence's importance. Verb's such as "said" do not make as much sense in a graph as verbs that have entities as the actor and the patient, such as "hired". In general, a good candidate will have two or more entities and a relation between them, and all relevant entities will be resolved. Pronouns or other unresolved entities are difficult to display on a graph.

Relations that were either incomplete or unlikely to be of interest on a graph were then filtered out of the list. A simple aggregate filter was used that combines a number of passes by filters that look at a single feature of a relation and decide whether to keep it. Only relations that passed through all of the filters were included in the graph. For this reason, filters that incorrectly removed too many relations were not included, while filters that removed only a few relations, but were highly accurate, were kept.

The filters target co-reference problems, as well as predicates that were likely mis-parsed. If a relation has an actors or patient consisting solely of a pronoun, it is removed from the final set to be displayed. Similarly, if it contains a word like this, that, these, or those it is also removed. These relations are likely referring to something outside the scope of the sentence, and since each sentence is parsed in a vacuum, there is no way to recover this information. There are tools that can solve co-reference problems, but that was outside the scope of this problem, and enough data could be extracted without attempting this difficult task that I chose to simply remove those sentences from the final set.

If a predicate was very long or very short, it was likely that the sentence had been mis-parsed. Some relations had a predicate that was only one character long, or did not exist at all. These usually came from sentences where the boundary was not found correctly. Either the sentence that was parsed actually contained two sentences, or only contained part of a sentence because the pre-processor split the sentence on a period that was part of an acronym instead of the end of a sentence. By filtering these types of sentences out I was able to compensate for that problem. Other relations were very long, which usually indicated a complex sentence where the predicate actually contained a conjunction or appositive or other grammatical structure making it unfit for display on a graph. It would be possible to identify and remove these from the dependency graphs that

C&C outputs, or even detect conjunctions and correctly output two relations, but that would require more in-depth analysis.

3.5 Visualization

The final list of relations was then used to create nodes for each of the unique actors, actions, and patients. It was also used to create directional links between each of them. To link some of the disjoint areas on the graph, the list of named entities longer than three characters in length was imported. The length minimum was due to the fact that some names, for example ”U. S.” produced the entities ”U”, and ”S”, which were present in a great number of actors and patients that had nothing to do with the United States. For all the remaining named entities, if the name occurred in any of the actors or patients, it was also added as a node, with links to all nodes that contained it.

The output from both the parser and the filter is used to generate XML with all the relations formatted for prefuse (Heer et al., 2005). It supports directed edges and multiple colors of nodes[1].

Blue nodes on the graph represent actors and patients extracted from an articles. Grey nodes represent predicates that link two actors and patients, and the lines between them contains arrows to show the correct direction to read the relation. In order to pull together multiple relationships about the same topic, named entities were displayed as red nodes on the graph, and were linked to any actors or patients that contained the same text. Some filtering was done to ensure that entities such as *U* were not contained in the set. This kind of entity was parsed from the data when an acronym was incorrectly split up because of punctuation, resulting in something like *U.N.* returning the named entities *U* and *N*. Since these shorter strings match many more other words than are actually related with them, a minimum length was used for named entities to be displayed on the graph.

4 Results

The training data for this project was gathered from Google News by searching for a topic and retrieving the text from several articles. For this evaluation, articles about Blackwater were used. The training set of data contains 150 sentences with over six thousand words. This data can be processed and a visualization can be created in under a minute. Blackwater was a word that was not likely to appear in articles about other topics, and the events referenced are all likely to be similar, providing good related material to pull relations from. This made it easy to find relevant articles.

Development data will be gathered in a similar fashion, and the results will be evaluated. News articles covering a different topic will be used to produce a graph. Statistics on the number of sentences that result in a relation, and the number and type of errors will be gathered. The final set of relations will then be scored by another person who will judge which make sense and are useful, and which do not. I hope to be able to find a relation for at least ten percent of sentences, and that of the relations I keep more than half are useful.

The final product allows an easy one-step transformation from article text to a graphical representation. The graphical representation does contain relevant information, and it is easy to read and navigate. The way multiple clusters are displayed does need to be altered to make them easier to read and navigate through. Currently, only one cluster can be displayed centrally, and the way the visualizer separates all connected nodes causes the remaining clusters to drift out of the field of view. It’s possible that something as simple as having one meta node connected to a central concept in each cluster would solve this problem.

The named entity recognizer output did not prove to be that useful. It was fairly reliable at finding people’s names, but it sometimes picked up other words, such as *school* or *of*. It frequently found parts of the noun phrase, but not the whole thing. Also, named entities were often not referred to that many times, especially not as part of a main noun phrase. Each topic was touched on briefly in an article, and only further references used pronouns, which are difficult to resolve properly and were not attempted for this project.

Some relations that were extracted did not have all of the information necessary to make them useful outside the context of the complete article. For example, several relations had only a pronoun for

[1]Modifications were made to the rendering code, which used the prefuse library, in order to support the different node colors

an actor, or contained words such as ”this”, ”that”, ”these”, and ”those”, which reference entities outside the sentence, and thus outside the scope of data the parser was given. There are tools that address co-reference problems such as these, but for this project I simply filtered those relations out. The goal is to retain as many relations as possible while filtering out the majority that are not useful. The filter code removed about one third of the relations, leaving the other two thirds to be displayed on the graph.

For example, the following relations have insufficient information, and should be excluded:

(3) we $\Longrightarrow$ want $\longrightarrow$ that person

(4) it $\Longrightarrow$ has built a $\longrightarrow$ fleet

(5) It $\Longrightarrow$ cited $\longrightarrow$ unnamed civilian and military officials

The format of these examples is ”actor$\Longrightarrow$predicate$\longrightarrow$patient”.

Other sentences were excluded that had features that were likely referencing something outside the scope of the sentence the relation was extracted from. For example, actors and patients that were pronouns were also excluded. I also exclude sentences with either very long or very short relations. Very long relations are both difficult to display in one bubble on the graph without intersecting with other nodes, as well as indicative of a mis-parsed sentence that contains some complex clauses that were not parsed properly. Relations consisting of less than two characters were usually indicative of a sentence where the boundary was not properly defined, and the predicate was identified as a period (.).

Some examples of good relations are:

(6) some officials $\Longrightarrow$ have expressed $\longrightarrow$ pessimism

(7) 25000 private contractors $\Longrightarrow$ protect $\longrightarrow$ diplomats

Some examples of excluded relations are:

(8) he $\Longrightarrow$ started $\longrightarrow$ the hearing

(9) Relatives $\Longrightarrow$ said they were on a family errand and posed $\longrightarrow$ no threat to the Blackwater convoy

(10) the new attorney general $\Longrightarrow$ makes $\longrightarrow$ this case

5 Conclusion

I was able to successfully create a graph of concepts and relations from news articles pulled from the internet with minimal human interaction. However, the graph is still too disjoint to be useful. More work needs to be done on getting more nodes connected, as well as extracting a higher percentage of data from the articles. It is possible to find useful information from the graph that is output though, and it is good at giving a general idea of the concepts presented in the articles.

From the original 150 sentences in the development set, 135 relations were produced by the parsing code. 43 of these made it through the filter to be displayed on the graph. The F1 score of the filter, as compared to human filtered relations, was .62.

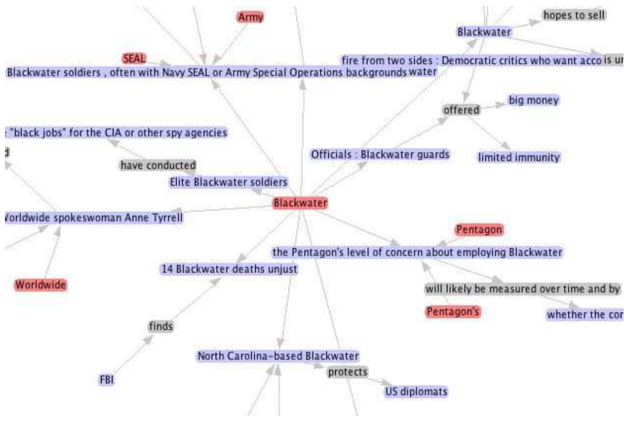
Army
hopes to sell
Blackwater
SEAL
fire from two sides : Democratic critics who want acco
is ur
Blackwater soldiers , often with Navy SEAL or Army Special Operations backgrounds
water
offered
big money
"black jobs" for the CIA or other spy agencies
Officials : Blackwater guards
limited immunity
have conducted
Elite Blackwater soldiers
Blackwater
Worldwide spokeswoman Anne Tyrrell
Pentagon
the Pentagon's level of concern about employing Blackwater
14 Blackwater deaths unjust
Worldwide
will likely be measured over time and by
Pentagon's
whether the cor
finds
North Carolina-based Blackwater
protects
FBI
US diplomats

Figure 5: Example of section of final output

Others have tackled the problem of aggregating multiple on-line sources (Mani and Bloedorn, 1997) as a graph. They used an algorithm to determine which nodes from each article should be considered the same node, and merged individual graphs to create one composite graph. This process could be used in conjunction to the solution that I have found to link multiple similar nodes.

Radev and McKeown (Radev and McKeown, 1998a) go one step further and recognize when two articles are talking about the same thing. They can extract similar sentences from each, giving them the ability to highlight agreement or disagreement. This could be combined with the results from this project to help cluster relations that cover the same thing.

6 Acknowledgments

I would like to thank Dr. Jason Baldridge for his guidance in the development of my application over the course of several semesters, including directing me to multiple useful techniques and tools without which this project would not have been nearly as successful. I would also like to thank both him and Dr. Raymond Mooney for their critique of the prior drafts of my thesis, as well as suggesting ways to evaluate my success. Additionally, I would like to thank Dr. Mohamed Gouda for serving on my thesis committee and providing feedback.

References

John Carroll and Ted Briscoe. High precision extraction of grammatical relations. *7th International Workshop on Parsing Technologies*, 2001.

Stephen Clark and James Curran. Formalism-independent parser evaluation with CCG and DepBank. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 248–255, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P07/P07-1032`.

Hercules Dalianis. SweSum - a text summarizer for swedish. *Technical report TRITA-NA-P0015, IPLab-174, NADA, KTH.*, 2000. URL `http://people.dsv.su.se/ hercules/papers/Textsumsummary.html`.

Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 121–128;, 1999.

Jeffrey Heer, Stuart K. Card, and James A. Landay. prefuse: a toolkit for interactive information visualization. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 421–430, New York, NY, USA, 2005. ACM. ISBN 1-58113-998-5. doi: http://doi.acm.org/10.1145/1054972.1055031.

Svetlana Hensman and John Dunnion. Using linguistic resources to construct conceptual graph representation of texts. *Lecture Notes in Computer Science*, 3206/2004:81–88;, 2004.

Julia Hockenmaier and Mark Steedman. Ccgbank: A corpus of CCG derivations and dependency structures extracted from the penn treebank. *Computational Linguistics*, 33(3):355–396, 2007.

Rao R. Lamping J. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages & Computing*, 7(1):33–55, 1996.

Inderjeet Mani and Eric Bloedorn. Multi-document summarization by graph search and matching. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 622–628;, 1997.

Paul Miller. Web 2.0: Building the new library. *Ariadne*, 2005.

Sougata Mukherjea. Information visualization for hypermedia systems. *ACM Comput. Surv.*, page 6. ISSN 0360-0300. doi: http://doi.acm.org.ezproxy.lib.utexas.edu/10.1145/345966.345984.

Jahna Otterbacher, Dragomir Radev, and Omer Kareem. News to go: hierarchical text summarization for mobile devices. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 589–596, New York, NY, USA, 2006. ACM. ISBN 1-59593-369-7. doi: http://doi.acm.org.ezproxy.lib.utexas.edu/10.1145/1148170.1148271.

Dragomir R. Radev and Kathleen R. McKeown. Generating natural language summaries from multiple on-line sources. *Comput. Linguist.*, 24(3):470–500, 1998a. ISSN 0891-2017.

Dragomir R. Radev and Kathleen R. McKeown. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24:470–500;, 1998b.