The Word Tree, an Interactive Visual Concordance

Martin Wattenberg and Fernanda B. Viégas

Abstract— We introduce the Word Tree, a new visualization and information-retrieval technique aimed at text documents. A word tree is a graphical version of the traditional "keyword-in-context" method, and enables rapid querying and exploration of bodies of text. In this paper we describe the design of the technique, along with some of the technical issues that arise in its implementation. In addition, we discuss the results of several months of public deployment of word trees on Many Eyes, which provides a window onto the ways in which users obtain value from the visualization.

Index Terms—Text visualization, document visualization, Many Eyes, case study, concordance, information retrieval, search.

1 Introduction

In James Joyce's A Portrait of the Artist as a Young Man, the word "his" appears 1,744 times. The word "her" occurs 316 times. These numbers provide little insight beyond a basic imbalance. Now consider that the most common word to follow "his" is "soul," while the most common word to follow "her" is "eyes." With this fact, the nature of the imbalance begins to emerge. Repeated elements tell us a great deal about texts—but with context more nuances and revealing themes appear.

Furthermore, the set of contextual elements often itself has a complex structure. "His soul" appears 83 times in A Portrait of the Artist as a Young Man, but what follows those two words? Among other phrases, "was fattening," "was festering," and "was foul"—along with "was waking," "was enriched," and "was soaring."

In this paper we introduce a new visualization technique, the word tree, that makes it easy to explore this type of repetitive context. A word tree places a tree structure onto the words that follow a particular search term, and uses that structure to arrange those words spatially. Simple interaction techniques allow the viewer to examine the ways that a particular word or phrase is used in a text, seeing broad patterns and drilling down into details.

The motivation for creating the word tree comes from our experience with user-generated visualizations on the Many Eyes site [11], which allows anyone to upload and visualize data. Since the site launched in the beginning of 2007, we have observed a growing number of attempts to visualize unstructured text. The first text visualization on the site was a tag cloud—a common technique showing word frequency. Despite the tag cloud's popularity, comments from users indicated they were sometimes as interested in usage context as raw frequency counts. The feedback prompted us to consider a technique that would retain more of the text composition for exploration.

The word tree was first made public in September 2007 and, as of March 2008, people have used the word tree to examine more than 650 texts. Because all activity on Many Eyes is public, and because many of its members write about their experiences on blogs, we have a rich record of word tree usage. We take advantage of this record to assess the word tree technique and examine the types of value that it is providing. We also describe feedback from users, which points to several promising research directions.

- *Martin Wattenberg is with IBM Research, E-Mail: mwatten@us.ibm.com.*
- *Fernanda B. Viégas is with IBM Research., E-Mail: viegasf@us.ibm.com.*

Manuscript received 31 March 2008; accepted 1 August 2008; posted online 19 October 2008; mailed on 13 October 2008.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

2 Related Work

Finding ways to display and refine search results is a classic challenge in information retrieval. The problem predates computers; for centuries biblical scholars have used concordances[1] to see how different words occur in religious texts. The computer-age equivalent of these paper concordances is the "keyword in context" (KWIC) technique [5], in which hits are shown with the search term, or keyword, surrounded by a snippet of the text in which it occurs. Although these snippets are often arranged so that the keywords are aligned, it can be difficult to see patterns and connections in the resulting array of text.

Several visualization methods have been proposed to show word usage. Tilebars [8] provide a compressed overview of search term distribution within a document; the SeeSoft program [4] provides graphical highlighting of textual metadata. The TextArc technique [13] shows the overall distribution of every word in a piece of text. None of these, however, provides an easy way to see the different contexts in which a given word or phrase is used. Other text visualizations, such as dotplots [7] and arc diagrams [14], spotlight global patterns of repetition but do not provide a detailed view of the context of usage of particular terms.

One clever technique for displaying contextual information about a search term is the "star diagram" of Bowdidge and Griswold [2]. Designed to help developers restructure code, a star diagram shows how a particular variable is used in a program. The display uses a tree structure to display which functions are applied to that variable, which functions call those functions, and so forth up the call stack. While it is not directly applicable to plain text, the star diagram's hierarchical arrangement of context makes it a precursor to the work we describe below.

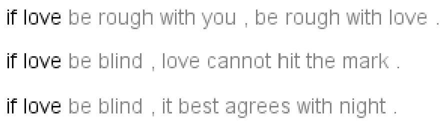
As we discuss in the next section, we propose to show context using an interactive tree structure, in which users can click on nodes to vary the level of detail. Three systems directly relate to the interaction techniques we employ. The network exploration program described in Yee et al. [15], the SpaceTree visualization [6], and Degree-of-Interest Trees (DOITrees) [9] use elegant animations to help users navigate, a design choice that we emulate. The SpaceTree and DOITrees, like our visualization, allow users to easily move up and down a hierarchy. On the other hand, our design for showing levels of detail and handling high branching factors contrasts with SpaceTree and DOITrees, and we discuss some differences in user reaction to this point.

[1] The word "concordance" is sometimes used to mean a comparison between texts. In this paper, however, we use it in the literary sense of an index that provides additional context for word usage.

1077-2626/08/$25.00 © 2008 IEEE Published by the IEEE Computer Society

3 Design of the Word Tree

A *word tree* is essentially an interactive form of the keyword-in-context (KWIC) technique. It builds on KWIC in three ways. First, it has a visual design that makes it easy to spot repetition in the contextual words that follow a phrase. Second, the design makes obvious the natural tree structure of the context. Third, it affords easy ways to explore the context further.



if love be rough with you , be rough with love .

if love be blind , love cannot hit the mark .

if love be blind , it best agrees with night .

Fig 1. All instances of “if love” in *Romeo and Juliet*.

A KWIC display can be thought of as a tree in disguise. Consider Figure 1, which shows the words that follow the search term “if love” in the play *Romeo and Juliet*. If one thinks of the search term as the root node, then the various distinct subsequent words define branches. In this case, because “if love” is always followed by “be,” it has just one child node, corresponding to the word “be.” The “be” node, however, has two children, one for each of the two distinct words that follow it: “rough” and “blind.” Continuing in this way one can define a tree structure that describes all the ways the search term is used.

This structure is not new. The basic idea, called a suffix tree, has for decades been an ingredient in string-processing algorithms. In fact textbook diagrams explaining suffix trees often resemble the word trees below. Despite their popularity among algorithm designers, however, suffix trees have not been used as a general visualization mechanism for search results.

3.1 Many Eyes as an experimental platform

Before describing the design of the word tree, we discuss some relevant features of our experimental platform, Many Eyes. The site allows anyone on the internet to try out different visualization techniques. Previous user interviews have found a broad set of backgrounds and goals among active participants on the site [3]. Making a new visualization method available on the site is an effective way to see how it will be spontaneously used by a diverse set of people.

Experimenting in public does add some constraints, however. Unlike a supervised deployment, we cannot rely on any training for the visualization. On the contrary, if people can’t rapidly make sense of what they see, they will probably click away to a different site. Our design, therefore, put a premium on simplicity and learnability.

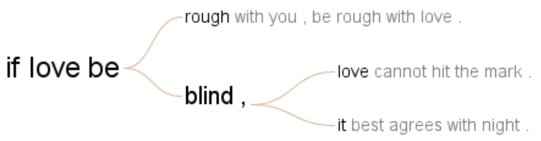

if love be
rough with you , be rough with love .
blind ,
love cannot hit the mark .
it best agrees with night .

Fig 2. Word tree showing all instances of “if love” in *Romeo and Juliet*.

3.2 Visual design

Figure 2 shows a simple example of a word tree, for the Romeo and Juliet example. The basic layout of the tree is a classic branching view. We chose this method for its instant readability. It immediately communicates to viewers that they are looking at a tree structure. Moreover, in contrast to more exotic methods such as hyperbolic trees or treemaps, it largely preserves the linear arrangement of the text.

Taking a cue from the popularity of tag clouds, we use font size to represent the number of times a word or phrase appears. The size is proportional to the square root of the frequency of the word. Using the square root rather than a linear scale achieves two goals. First, it means the area of the word is very roughly proportional to the frequency (except for variations created by word length). Second, it leaves sufficient blank space above and below that the overall tree structure is visually obvious.

Branches of the tree continue at least until they define a unique phrase used exactly once. Instead of stopping at the first unique phrase, the tree continues until a period is reached (up to a fixed limit of tokens), so that viewers see sensible fragments of the text. To distinguish between the main tree of unique phrases, and the additional context, the former is colored black and the latter is drawn in gray.

One somewhat counterintuitive design choice is that we do not discard stopwords or even punctuation. The rationale is that prepositions and commas are often critical to understanding the meaning of a text. Leaving them out might put together phrases that mean very different things. As we discuss later, this has proved controversial with our users.

3.3 Interactivity

To start exploring a word tree visualization, the user types a word or phrase into a “search” box at the top of the screen. Each time the user types the “enter” key, a punctuation mark, or a space, the tree is rebuilt. This allows a responsive feel, and for multi-word phrases lets the user quickly see if initial words in the phrase have too few hits to be worth typing additional words. Note that it would also be possible to build the tree letter by letter; however, early experiments suggested this would be distracting.

Once a word tree is shown, a user can interact with it. Moving the mouse over a particular word or phrase brings up additional information, along with a message saying that clicking will explore the tree further. Clicking on an individual word will redefine the phrase shown by the tree. This can either narrow or widen the text search. For instance, if the current phrase is “if love,” clicking on the initial “if” will re-center the tree on the phrase “if” (see Fig. 3A). On the other hand, if the user clicks on a word in a branch of the tree, such as “blind” in the branch “if love be blind,” then the tree will be re-centered on the longer phrase, “if love be blind” (Fig. 3B).

Often when looking at a tree, a user will see an unexpected or interesting word in a branch, and may want to see all uses of that word. To support this goal, a second click option is control-click, which recenters the tree on the single word clicked, no matter where it occurs in the tree. Control-clicking on “blind” in the example above will display a tree that is rooted on the word “blind” (Fig. 3C).

Someone encountering the word tree for the first time may find the result of clicking or typing unexpected. This issue is especially acute for visualizations on Many Eyes, since people can first see these visualizations embedded on totally unrelated websites, without any explicit instructions. To clarify what is changing in the tree when the user clicks or types, we built in animated transitions when possible. For example, if the user types “if love” into the search box and hits “enter,” they see the following:

1. The tree is built when the space after “if” is typed, fading gently into place to avoid an abrupt “flash” of information.
2. After “enter” is typed, an animation occurs, in which the branch for “love” becomes bigger and the other branches fade away.

Additional adjustments occur automatically. The text scale changes to put as many words as possible on the screen, while making sure that the largest words are readable. For repetitive texts, the word tree can sometimes take a large amount of horizontal space, so scrollbars are provided, Overall, the fluid feel of the interaction is similar to that of the radial tree explorer discussed in Yee *et al*. [15] and the SpaceTree system [6].

One option is to show context words trailing the given search phrase, another option is to show words that precede that phrase. Switching between these options is not animated, since there is not a sensible way to interpolate between them. Another free parameter in the visualization is the order of the branches beneath each node. The

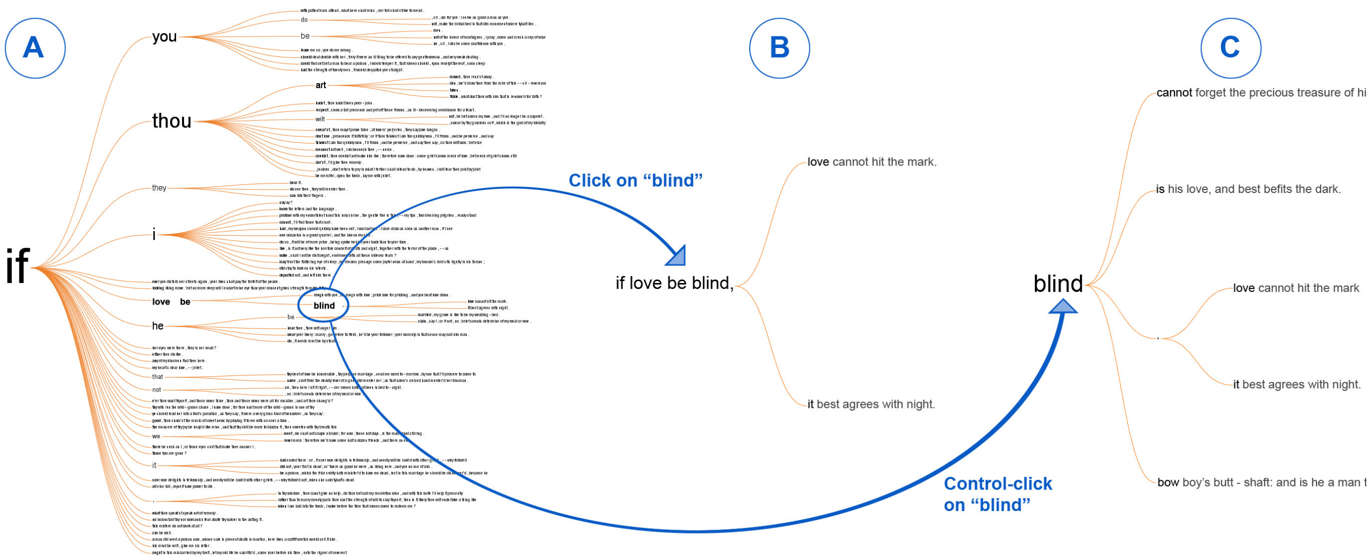

A
B
C
if
you
do
be
thou
art
wit
they
i
love be
blind
he
that
not
we
it
Click on “blind”
if love be blind,
love cannot hit the mark.
it best agrees with night.
Control-click on “blind”
blind
cannot forget the precious treasure of hi
is his love, and best befits the dark.
,
love cannot hit the mark
it best agrees with night.
bow boy's butt - shaft: and is he a man t

Fig 3. Sequence showing some of the interaction options in the word tree. In figure A, the user has typed the word “if” in *Romeo and Juliet*. In B, the user has clicked on “blind,” which appears in one of the branches under “if.” This causes the visualization to recenter to the longer phrase “if love be blind.” In C, the user Control-clicks on “blind,” which causes the visualization to recenter to blind by itself, revealing that there are additional phrases after this term.

Many Eyes word tree provides a choice among three options. The branches can be arranged alphabetically (making it easy to scan for particular words), by frequency (so the largest branches are first), or by order of first occurrence in the text (the default option, since it often produces a tree that best reflects the underlying text.) As with clicking, when the user switches between two of these options the word tree animates smoothly to help make clear what is changing.

As the user interacts with the tree—she may click on a branch, recenter the tree, choose a different search term, etc.—the word tree tracks of the sequence of actions just as a web browser does. This allows the user to click on browser-like “back” and “forward” buttons to review her previous steps in the visualization. This feature helps users quickly switch between desired states for comparisons and easily retreat from navigational dead ends.

As with all visualizations on Many Eyes, users can set particular states and make comments. In doing so, they may wish to point to particular items on the visualizations. To support this, users can set the visualization to a “highlighter mode,” where clicking on words will not cause a recentering of the tree, but instead highlight words with translucent brown circles. Thus a user can leave a comment like, “Note the position of God in this context,” and highlight “God” so that other readers do not need to search for where it occurs.

Finally, the word tree does not provide any sort of “overview” of the text nor does it present an initial search term for viewers to start from. In this way, the visualization resembles an information retrieval interface, driven by a search term rather than starting with an overview. The reason for this design choice is that without a search term, there is no obvious entry point—several alternatives with suffix-tree-like beginnings were attempted, but seemed busy and uninformative. A future version might try to automatically find a good starting point: perhaps a tree centered on the most frequent terms, a tree that shows the highest number of separate branches, or a tree with the deepest branches. Having a default start point might solve certain problems. For instance in the current system, unless the creator of the word tree actively sets an initial search term, the visualization will look blank to subsequent viewers on the site. Another limitation of not having an overview is that users need to know a bit about the underlying data to make sure that they look for words that appear in the text. Many other interactive features are possible. We discuss these in the sections on user feedback and future work.

4 Implementation Considerations

The current implementation of the Word Tree on Many Eyes is a Java applet, written using JDK 1.4. It is engineered to handle texts with up to 1,000,000 tokens. (In addition to being a pleasingly round figure, this is the approximate number of tokens in the King James Bible, probably one of the most-visualized text on Many Eyes.) In this section we discuss some of the implementation details and decisions that allow the applet to scale—both visually and in performance—to a million tokens.

The data structure behind the word tree—that is, the hierarchical structure of the context words—is well-known to computer scientists as a “suffix tree.” In our context the practical bound on performance is memory rather than CPU cycles: constructing the tree is fast (at least for a million-token text) as long as there is sufficient memory. Java applets often have limited heap space, as low as 64MB. Although this may seem more than adequate for holding a million-node tree, it is actually a serious constraint due to the memory-intensive nature of Java objects. To get around the problem, we do not create a suffix tree for the entire text, but rather create the suffix tree on the fly, a new one for each phrase typed in. In practice this saves a significant amount of memory; for instance, in the King James Bible (about 1,000,000 tokens), the word tree for “the” has only about 64,000 leaves. This complicates effects such as animated transitions, but permits the feeling of instant feedback we desire.

In addition to the data-level scaling, two issues arise in scaling the tree visually. The first is that the total number of branches is huge compared to the screen size. When there are tens of thousands of leaves to a tree, there is no sensible way of displaying all of these on a screen that is a few hundred pixels high. We resolve this issue by a standard “level of detail” method. As the geometry of the tree is defined, when it is determined that a subtree takes up less than 3 pixels of vertical space, we do not draw the entire subtree. Instead, we find the deepest branch, and draw that. By doing so, we show the overall shape of the tree, but do not draw more than necessary. This simplifies the display and also keeps the number of rendered objects low enough that smooth animated transitions are possible.

One might contrast this with the method used by Bederson et al [6] in the SpaceTree. In that visualization, once items become too small to see, subtrees are replaced by icons or simplified views that indicate the overall breadth and depth. Note that our method does not need to communicate a general sense of breadth, since we only apply level of detail calculations when the breadth of a subtree is negligible compared to the breadth of the overall tree.

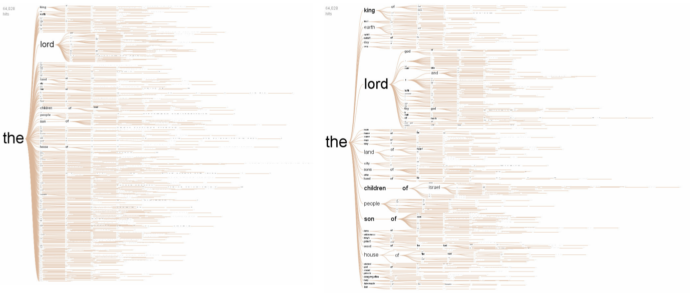


Fig 4. Two versions of the word tree: on the left, all branches under "the" are displayed, causing most words to become unreadable. On the right, we use a "level of detail" method, showing a subset of branches.

A second design issue arises for common words in a large text: for instance, in the word tree for "the" in the King James Bible, which occurs 64,028 times. 3,604 distinct words follow "the" in this text, with the most common being "Lord" at 11% of the total. That is enough to be visible, but the second most common word, "son," takes up only 2.3%, barely enough to be legible. As a result, if all words were shown—or even if only a few were shown with sizes proportional to their frequency—then almost none would be readable.

To handle this problem, we compromise and show only the largest branches, with the number defined so that only branches with at least 1% of the total leaves are included. In the case of the King James Bible and "the," this means that words like "son," "children," and "king" are legible. This method of pruning is continued recursively for sub-branches. Note that the result is a genuine compromise, since a user can't immediately deduce the true proportional usage of a word in context, since we have removed the "long tail" of infrequently seen words. It is worth emphasizing the difference from the kind of data used in SpaceTree, where the prototypical use case was an organizational chart with branching factors in the dozens, not thousands. An approach similar to our is taken by the DOITree described in [9], which does collapse nodes of lesser significance to make room for others; on the other hand—but as with the SpaceTree, the DOITree strives for readability of individual items overs a sense of overall breadth,

5 Spontaneous Usage on Many Eyes

As of this writing, users have created 658 word trees on Many Eyes. This section describes some of these word trees and the types of data they have been used for[2]. Because of the ease of access to Many Eyes, some of these word trees represent undirected "tire-kicking" and aimless experimentation. In other cases, however, users had specific goals in mind and described their actions in detail. We found these more detailed cases through two avenues: on and off site. User comments and actions on the site led us to several examples. We also performed searches on Google, which reported more than 300 references "word tree" with either "manyeyes" or "many eyes" outside of the ibm.com domain.

The search led us to some of the most detailed reports. In one, entitled "Using Word Tree Visualization for Checking Title Consistency" [1], a blogger wrote a 1,007-word essay describing his use of the visualization. His initial task was creating a series of title-like summaries for stories in the Bible. He decided to use the word tree to visualize his collection, and even went to the trouble of adding special "+Start+" and "+end+" tokens to his titles so that they would not run together in the tree. After doing this, he reported:

Looking at the frequency-sorted suffixes for "+start+ Jesus warns", i see a large group under "against", and a number under "about", but also a single instance, "Jesus warns of coming judgment". Because the third word is "of" rather than "about", it stands apart from the other instances which really share the same concept.

He went on to describe how he could rewrite the title to become more consistent. Along with describing the value he derived from the visualization, he also provided a very detailed description of the word tree and how he interacted with it.

... clicking on "teaches" narrows the view further (which you pretty much have to do to see the details: results over 30 or 40 aren't really visible). One advantage of this representation is that it gives you some help in knowing what to explore (in user interface terminology, an affordance). Though i can't see all the details without zooming in, i can see a significant cluster of titles starting with "Jesus warns", and if that's interesting, i can click on "warns" to zoom in and see those 18 titles.

This description of the effect of the too-small-to-read type is very interesting, because it stands in contrast to a result reported for the SpaceTree. In [6], Grosjean et al. emphasize that their users "rejected bluntly" unreadable type, and the authors of the paper created a new type of icon as an alternative to scaled-down text. It is unclear whether this blogger would have preferred the SpaceTree icons; it is also possible that there is great variability among individual users' preferences, or that the different branching factors in the SpaceTree data versus our suffix trees are amenable to different designs.

Finally, this blogger also made a suggestion for a future feature, namely drilling down for more details:

What would be really great would be to turn this from a visualization into a navigation system, so once i've drilled down to "Jesus warns against ...", then i could select a title and actually view the pericope text.

While this essay-length blog entry was unusual, the creativity and energy behind it were echoed in the actions of many other users. We now describe some of the general trends in how they used the word tree.

5.1 Visualizing the spoken word

In April of 2007, former U.S. Attorney General Alberto Gonzales testified before the Senate Judiciary Committee on his role in the dismissal of several U.S. attorneys. As the transcript of Mr. Gonzales's testimony became available on the web, one of the authors in this paper visualized the text as a word tree on Many Eyes. The visualization clearly showed a number of times when Mr. Gonzales had used expressions such as "I don't recall," "I don't know," "I don't think" while testifying (Fig. 5).

In September, the Gonzales visualization was featured on the front page of Many Eyes—a prominent spot on the site. Within 90 minutes of it being featured, another user had created a new word tree, entitled "William 'I don't recall' Jefferson Clinton Testimony in Sexual Harrassment Lawsuit that led to his impeachment" [sic]. Here

[2] In this section we reprint some visualizations created by registered users of Many Eyes. As part of registration, all users gave permission for us to create such copies.

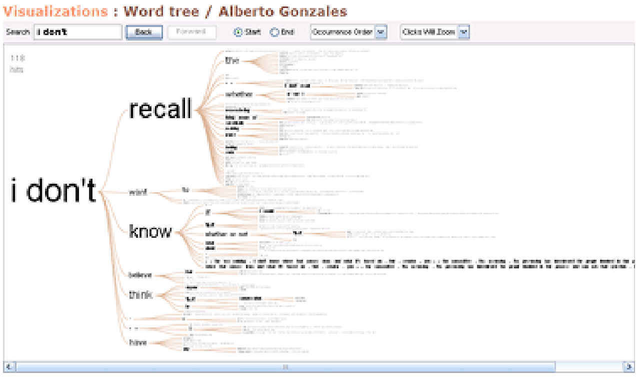


Fig 5: Alberto Gonzales' testimony in 2007.

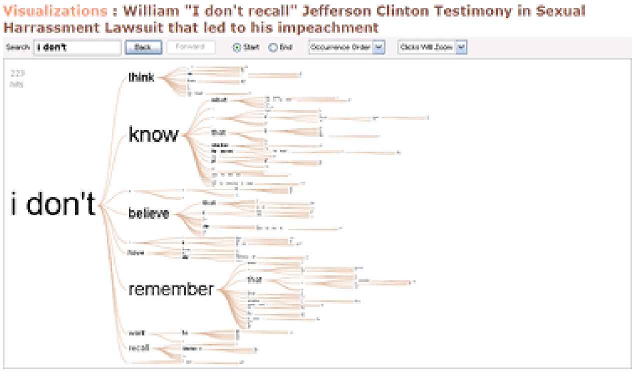


Fig 6: Bill Clinton's testimony in 1998.

too, the user set up the visualization so that the default view would show the number of times Mr. Clinton had said "I don't know," "I don't remember," "I don't think," etc (Fig 6). In short, both visualizations gave a clear portrait of evasive testimony.

Given that these scandals focused on politicians at opposite ends of the political spectrum, the visualizations take on an evident spin, with even the act of their creation suggesting political affiliations and beliefs. This sort of contribution to "counter" someone else's creation on Many Eyes indicates that users are integrating these tools in their communicative practices. Far from being dispassionate representations of data, the two "I don't recall" word trees are part of a political conversation, a dialog happening through visualization.

The ability to visualize political transcripts has resonated with our user base. Since the word tree was launched during the preparation for the 2008 U.S. presidential election, users frequently created word trees of political speeches, debates among candidates, and media coverage of the election.

Emotionally charged transcripts such as congressional hearings and political speeches are not the only kind of transcripts being visualized on Many Eyes. Even communities that are traditionally immersed in numerical data, such as financial analysts and investors, have started to explore the possibilities of using word trees to visualize transcripts. Earlier this year Seeking Alpha, a well-known online column of stock market opinion and analysis, embedded both a word tree and a tag cloud of the transcript of an earnings conference call on its site and invited readers comment on their value.

The post quickly generated a number of comments, not all of them in approval of the experiment. Some felt that the word tree was more helpful than the tag cloud because it kept the structure of the text, while others mentioned that it was easier not to use visualization at all:

Just give us the text, we know how to find (Ctrl+f)

As with the Blogos author, a common request was for the ability to click on an item in the visualization and see the places in the raw transcript where that item appears.

5.2 Visualizing the written word

The word tree was designed to handle texts of up to a million tokens, and to demonstrate this we created a visualization of the King James Bible, which contains 1,007,116 words and punctuation marks. Once the visualization was posted on the site, it was quickly picked up by a group of users interested in religious texts. The reaction was positive; this comment, unusual for visualizations in general, typified the response:

This is a new tool to teach the Bible's truth. God bless you.

Other users promptly explored various entryways into the text, looking for expressions such as "days of thy," "my love," and "love the lord" (Figure 9). As previously noted [13], visualizations of religious data have been a regular occurrence in Many Eyes since the site was launched. Perhaps it is not surprising that this community would be excited to experiment with the analytical possibilities of the word tree.

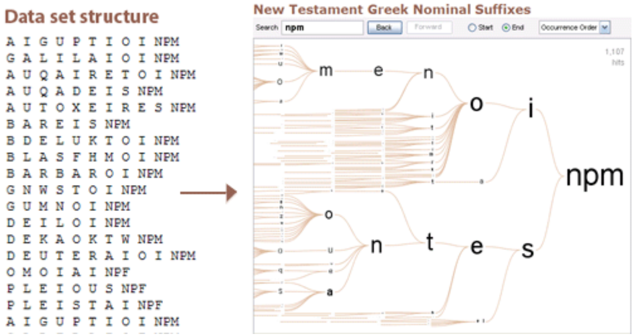
Users have also created numerous word trees of literary works, musical lyrics, and academic papers. An interesting trend is the visualization of online social activity. Some users have started visualizing collections of Twitter posts, blog posts, and newsgroup discussions. It seems that, like tag clouds, word trees might be helpful in giving people a quick sense of distributed activity online.

5.3 Visualizing structure

Although the word tree was designed to analyze unstructured text, it is based on a visualization of abstract tree structures. Users quickly caught on to the possibility of visualizing structured data and started specially formatting data in ways that would induce the word tree to show tree-structured information.

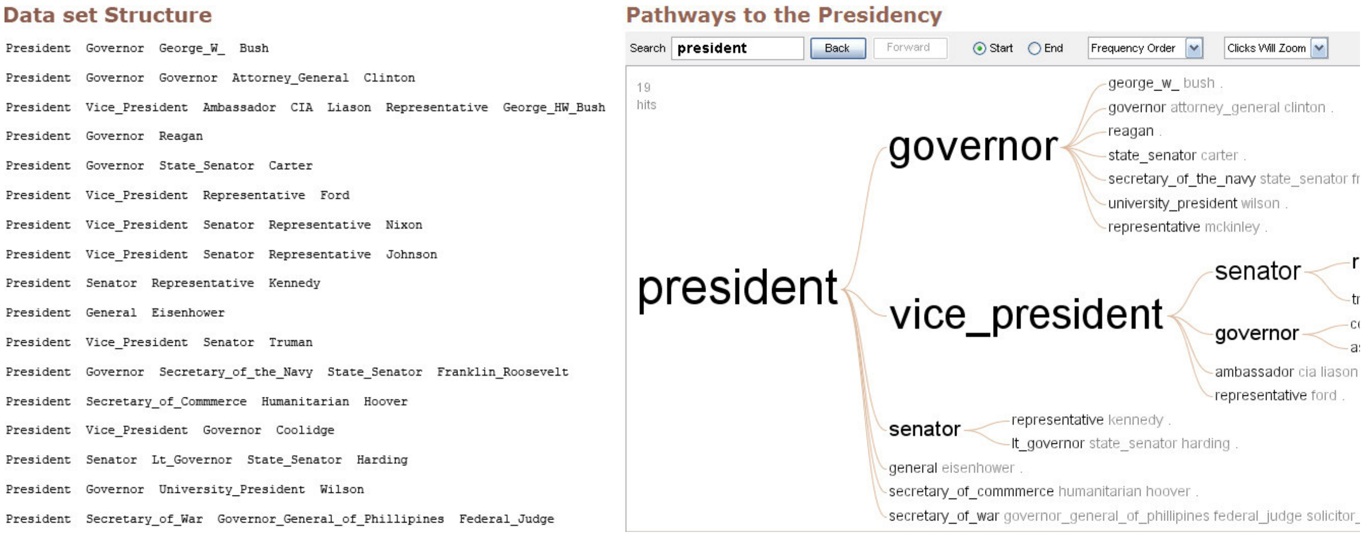
One person uploaded a data set of Greek nominal suffixes used in the New Testament with full nominal morphology. Because this data set is not a regular text passage but rather a list of words spaced out into individual letters, the word tree looks cryptic (see Fig. 7). If, for example, a user does a search for, NPM (nominative, plural, masculine words), they will see the suffix tree is dominated by –OI and –ONTES. This arrangement shows that the large majority of nominative, plural, masculine words in Greek end in -OI or -ONTES.

Another user created a data set to show the different pathways to the U.S. Presidency. The data set lists the names of 19 American presidents and the sequence of titles held by each one of them (Fig.


Data set structure

```
A I G U P T I O I NPM
G A L I L A I O I NPM
A U Q A I R E T O I NPM
A U Q A D E I S NPM
A U T O X E I R E S NPM
B A R E I S NPM
B D E L U K T O I NPM
B L A S F H M O I NPM
B A R B A R O I NPM
G N W S T O I NPM
G U M N O I NPM
D E I L O I NPM
D E K A O K T W NPM
D E U T E R A I O I NPM
O M O I A I NPF
P L E I O U S NPF
P L E I S T A I NPF
A I G U P T I O I NPM
```

Fig 7. Data set and word tree of Greek nominal suffixes in the Bible. Here, "npm" refers to nominative, plural, masculine nouns.



Data set Structure

```
President  Governor  George_W_  Bush
President  Governor  Governor  Attorney_General  Clinton
President  Vice_President  Ambassador  CIA  Liason  Representative  George_HW_Bush
President  Governor  Reagan
President  Governor  State_Senator  Carter
President  Vice_President  Representative  Ford
President  Vice_President  Senator  Representative  Nixon
President  Vice_President  Senator  Representative  Johnson
President  Senator  Representative  Kennedy
President  General  Eisenhower
President  Vice_President  Senator  Truman
President  Governor  Secretary_of_the_Navy  State_Senator  Franklin_Roosevelt
President  Secretary_of_Commmerce  Humanitarian  Hoover
President  Vice_President  Governor  Coolidge
President  Senator  Lt_Governor  State_Senator  Harding
President  Governor  University_President  Wilson
President  Secretary_of_War  Governor_General_of_Phillipines  Federal_Judge
```

Pathways to the Presidency

Search president
Back
Forward
● Start ○ End
Frequency Order
Clicks Will Zoom
19 hits
president
governor
george_w_ bush .
governor attorney_general clinton .
reagan .
state_senator carter .
secretary_of_the_navy state_senator fr
university_president wilson .
representative mckinley .
vice_president
senator
r
tr
governor
co
as
ambassador cia liason
representative ford .
senator
representative kennedy .
lt_governor state_senator harding .
general eisenhower .
secretary_of_commmerce humanitarian hoover .
secretary_of_war governor_general_of_phillipines federal_judge solicitor_

Fig 8. Data set and word tree of pathways to the U.S. presidency.

8). The word tree reveals that, whereas there were ten presidents who had served as governors, only five had previously been senators—Harding was the first and Kennedy the last. One might wonder what this historical fact spells for the 2008 elections.

Among other things, these examples show that at least some users can understand how the data structure maps to the visualization technique well enough to reverse-engineer the visualization to serve new purposes.

5.4 User feedback

Working on a public web site for visualization gives us the ability to collect user feedback on the new techniques we launch. The word tree has generated many comments about what users found helpful, problematic, and even missing. The response has generally been positive with a number of suggestions for improvements.

Users were excited about the ability to visualize large pieces of text while keeping some of the context around keywords. Several people remarked on how much they liked the capability of “zooming” in and out of specific branches of the tree. When comparing the word tree to the tag cloud, users felt that the word tree allowed them to engage in deeper analysis of the text.

On the other hand, not all feedback was positive and users suggested many new features. Several of these are relatively minor changes. The most commonly requested feature was for an option to ignore punctuation, and sometimes stopwords as well. Because sometimes these tokens take a lot of room in the visualization, users would prefer to have the option of turning stop-words and punctuation on and off. In addition, some users also wanted more context, in particular the ability to place their search terms “in the middle” of the word tree, surrounded by both the preceding and following context a given phrase.

The second main group of suggestions centered on the ability to drill down from the word tree into a plain view of the text. A related request is to see the locations of all the uses of a particular word or phrase, perhaps by drawing lines from the nodes of the word tree into a vertical line representing the extent of the text. Finally, several users requested the ability to filter the text, showing word trees of just particular sections.

CONCLUSION AND FUTURE WORK

The Word Tree is sufficiently flexible and engaging that hundreds of people have used it to examine data on the Many Eyes site. The technique presented here is extremely simple, however, and there are many natural extensions. We note some promising future directions in this section, and then conclude with a brief discussion of what the popularity of the Word Tree may tell us about visualization of text on the web.

As described above, users have provided a long list of desired features. Many of these, such as drilldown from the tree into the original text, are straightforward to implement, but a few point to larger research directions. One kind of implicit request from our users regards comparisons. Many users like to look at different texts, or different phrases in the same text. For example, people will create word trees of different sections of the Bible, or might look at the context for contrasting words like “his” or “her” and what follows them. To support this type of comparison, it would be nice to overlay two trees on top of one another, perhaps with some sort of color coding.

A second idea, suggested by several users including [10] is to show a “net” of the words that connect two phrases. In other words, a user could type in “Romeo” and “Juliet” when studying Shakespeare, and see all chains of words (of less than a given length) that connect the two. These chains would not form a tree, of course, but a sort of network anchored at the two ends. Exactly how best to display and interact with such a net is an interesting problem.

Another natural direction is to handle much larger data sets. Doing so would probably require an offline calculation of a suffix tree, which would be stored on a server. Algorithms for doing such computations exist—they are handy in biology, for example—but may have to be modified to handle level-of-detail issues.

One problem identified users is that there is no natural entry point into the visualization: the viewer starts by seeing a blank screen. It might be worthwhile combining a word tree with other text visualizations, whether a tag cloud or a more complex system such as Jigsaw [12] that could provide an initial analysis of entities of interest.

Finally, the hundreds of word trees created on the Many Eyes site point to a broader implication: people are hungry for new ways to look at unstructured data. Predicting the exact use cases for these visualizations is difficult. Before seeing the first bible visualizations on Many Eyes, for instance, we would not have guessed at the popularity of religious analyses. Given the broad demand for text visualizations, however, it seems like a fruitful area of study.

Acknowledgements

The authors thank Frank van Ham, Jesse Kriss, Matt McKeon, Lee Byron, and Eric Gilbert for helpful suggestions. In addition, we are grateful to the users of Many Eyes for their creativity and willingness to provide feedback on an experimental visualization technique.

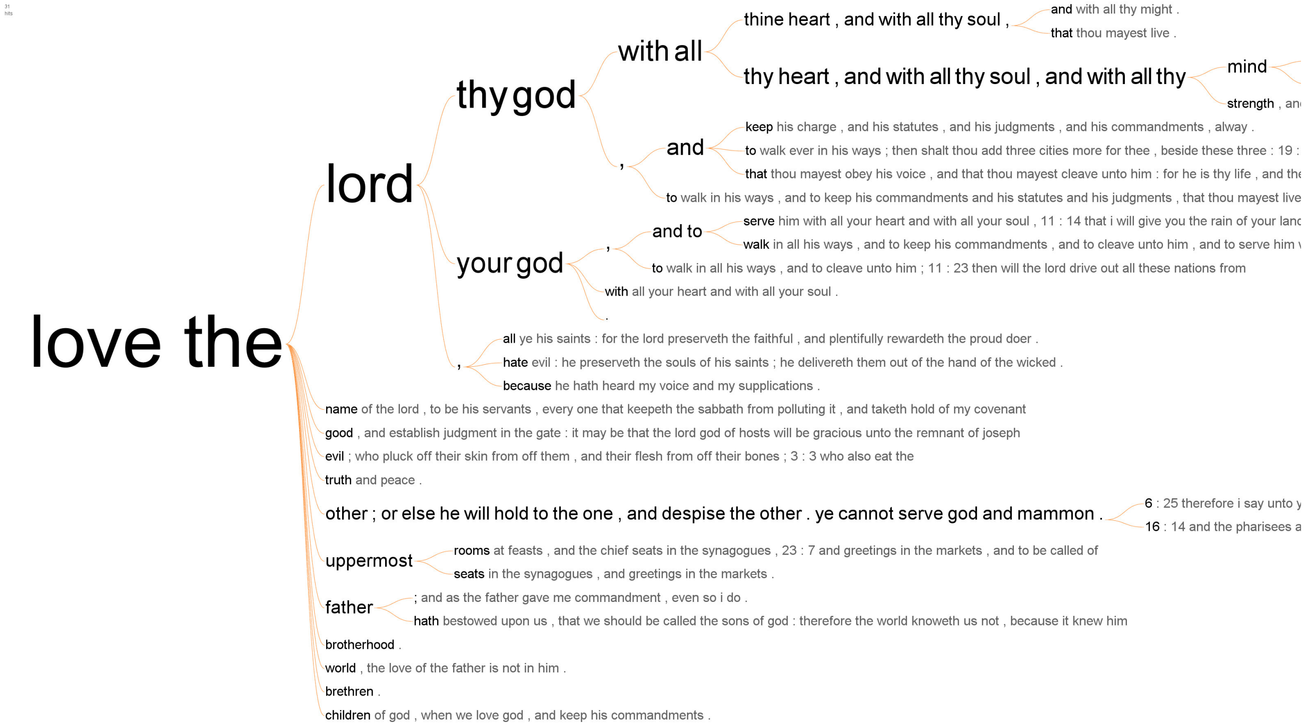

31 hits
love the
lord
thy god
with all
thine heart , and with all thy soul ,
and with all thy might .
that thou mayest live .
thy heart , and with all thy soul , and with all thy
mind
strength , an
,
and
keep his charge , and his statutes , and his judgments , and his commandments , alway .
to walk ever in his ways ; then shalt thou add three cities more for thee , beside these three : 19 :
that thou mayest obey his voice , and that thou mayest cleave unto him : for he is thy life , and the
to walk in his ways , and to keep his commandments and his statutes and his judgments , that thou mayest live
your god
,
and to
serve him with all your heart and with all your soul , 11 : 14 that i will give you the rain of your land
walk in all his ways , and to keep his commandments , and to cleave unto him , and to serve him
to walk in all his ways , and to cleave unto him ; 11 : 23 then will the lord drive out all these nations from
with all your heart and with all your soul .
.
,
all ye his saints : for the lord preserveth the faithful , and plentifully rewardeth the proud doer .
hate evil : he preserveth the souls of his saints ; he delivereth them out of the hand of the wicked .
because he hath heard my voice and my supplications .
name of the lord , to be his servants , every one that keepeth the sabbath from polluting it , and taketh hold of my covenant
good , and establish judgment in the gate : it may be that the lord god of hosts will be gracious unto the remnant of joseph
evil ; who pluck off their skin from off them , and their flesh from off their bones ; 3 : 3 who also eat the
truth and peace .
other ; or else he will hold to the one , and despise the other . ye cannot serve god and mammon .
6 : 25 therefore i say unto y
16 : 14 and the pharisees a
uppermost
rooms at feasts , and the chief seats in the synagogues , 23 : 7 and greetings in the markets , and to be called of
seats in the synagogues , and greetings in the markets .
father
; and as the father gave me commandment , even so i do .
hath bestowed upon us , that we should be called the sons of god : therefore the world knoweth us not , because it knew him
brotherhood .
world , the love of the father is not in him .
brethren .
children of god , when we love god , and keep his commandments .

Fig 9. Word tree of the King James Bible showing all occurrences of “love the.”

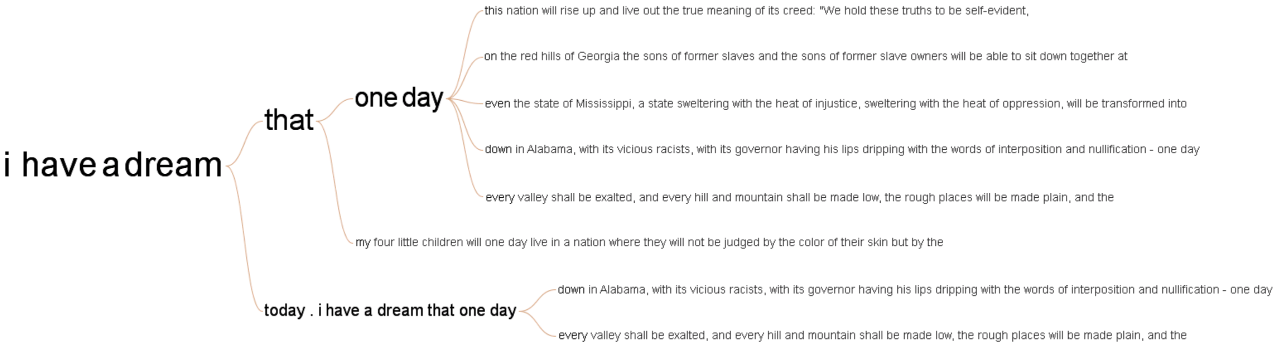

i have a dream
that
one day
this nation will rise up and live out the true meaning of its creed: "We hold these truths to be self-evident,
on the red hills of Georgia the sons of former slaves and the sons of former slave owners will be able to sit down together at
even the state of Mississippi, a state sweltering with the heat of injustice, sweltering with the heat of oppression, will be transformed into
down in Alabama, with its vicious racists, with its governor having his lips dripping with the words of interposition and nullification - one day
every valley shall be exalted, and every hill and mountain shall be made low, the rough places will be made plain, and the
my four little children will one day live in a nation where they will not be judged by the color of their skin but by the
today . i have a dream that one day
down in Alabama, with its vicious racists, with its governor having his lips dripping with the words of interposition and nullification - one day
every valley shall be exalted, and every hill and mountain shall be made low, the rough places will be made plain, and the

Fig 10: Word Tree showing all occurrences of “I have a dream” in Martin Luther King’s historical speech.

REFERENCES

[1] Boisen, Sean, “Visualizing Bible Data at Many Eyes.” Blog entry. http://semanticbible.com/blogos/2007/01/25/visualizing-bible-data-at-many-eyes/

[2] Bowdidge, R. and Griswold, W. (1998) Supporting the restructuring of data abstractions through manipulation of a program visualization. *ACM Transactions on Software Engineering and Methodology.* 7(2) 109-157.

[3] Your Place or Mine? *Visualization as a Community Component*. Catalina M. Danis, Fernanda B. Viégas, Martin Wattenberg, Jesse Kriss.CHI, 2008.

[4] Eick, S., Steffen, J., and Sumner, E. (1992) Seesoft-A Tool for Visualizing Line Oriented Software Statistics. *IEEE Transactions on Software Engineering*, 18(11), pp. 957-968.

[5] Fischer, M. (1966). The KWIC index concept: A retrospective view. *American Documentation*. 17 (2) pp. 57 -70

[6] Grosjean, J., Plaisant, C., Bederson, B. (2002). SpaceTree: Supporting Exploration in Large Node Link Trees, Design Evolution and Empirical Evaluation. *IEEE Symposium on Information Visualization*.

[7] Helfman, J. (1996) Dotplot patterns: a literal look at pattern languages. *Theory and Practice of Object Systems*, 2(1) pp. 31 – 41.

[8] Hearst, M. (1995) TileBars: Visualization of Term Distribution Information in Full Text Information Access, *ACM Conference on Human Factors in Computing Systems.*

[9] Heer, J. and Card, S. (2004) DOITrees Revisited: Scalable, Space-Constrained Visualization of Hierarchical Data. *Proc. Advanced Visual Interfaces*

[10] Hurst, M. “Word Trees.” Blog entry. http://datamining.typepad.com/data_mining/2007/09/word-trees.html

[11] Paley, B. (2002) TextArc. http://www.textarc.org

[12] Stasko, J., Gorg C., and Liu, Z. "Jigsaw: Supporting Investigative Analysis through Interactive Visualization", Information *Visualization*, Vol. 7, No. 2, Summer 2008, pp. 118-132

[13] Viégas, F.B., Wattenberg, M., van Ham, F., Kriss, J., & McKeon, M. (2007) Many Eyes: A Site for Visualization at Internet Scale. *IEEE Symposium on Information Visualization*.

[14] Wattenberg, M. (2002) Arc Diagrams: Visualizing Structure in Strings. *IEEE Symposium on Information Visualization*.

[15] Yee, K, Fisher, D., Dhamia, R., and Hearst, M. (2001) Animated Exploration of Graphs with Radial Layout. *IEEE Symposium on Information Visualization*