Visualizing Links between Sentences in Asynchronous Conversations: a New Feature for ConVis

Jordon Johnson, UBC (jordon@cs.ubc.ca)

Abstract— Asynchronous conversations are becoming increasingly commonplace with the continuing proliferation of email, blogs, forums, and other discussion websites. The exploration, analysis and summarization of these conversations are thus an area of interest to natural language processing (NLP) researchers. Researchers at UBC have developed ConVis, a state-of-the-art visualization tool that facilitates exploration and analysis of asynchronous conversations. One concept that may prove useful in these endeavours is that of referential relationships, or **links**, between sentences. I present an extension to ConVis that allows users to carry out three tasks related to exploring and annotating these links. The most immediately applicable of these tasks is the improvement of an existing gold standard dataset, where a number of the existing link annotations are found to be in need of revision.

Index Terms—natural language processing, information visualization, ConVis, asynchronous conversations, linking sentences, arc diagrams

INTRODUCTION

Asynchronous conversations are ubiquitous on the internet today, from email conversations to discussions on websites such as Reddit and Facebook. In an asynchronous conversation, users can post **comments** (hereafter also known as **posts**) in response to existing posts; and the initial post may be an **article** as in the case of many blogs and news websites. These conversations naturally form a tree structure, where each node is a post, and its children are the posts that respond directly to it. This tree structure removes many of the timeline constraints inherent in the more traditional synchronous conversations by allowing participants to contribute posts at almost any point in the conversation. While some temporal constraints are still maintained—a given post's children are timestamped and ordered temporally—these constraints also reinforce the asynchronous nature of the conversation as a whole, since parts of a branch displayed higher in the conversation may be more recently posted than those further down. Due to their proliferation and structural differences from synchronous conversations, asynchronous conversations are a significant topic of study to NLP researchers.

Of particular relevance to this work is the concept of referential links between sentences—the idea that some sentence in one post refers to a particular sentence in an ancestor post (see Figure 1). Note that there is a temporal constraint inherent in a link; a sentence cannot refer to something not yet written, and so link directionality is inherently "upwards", assuming proper use of the discussion website in question. These links may prove to be of use in analysing asynchronous conversations. For example, if many sentences in an asynchronous conversation link to a particular sentence in its initial article, then that sentence is more likely to form part of a summary of that conversation than other sentences with fewer links. In addition, heavily linked sentences could be of interest to journalists and researchers interested in learning what makes some sentences spark more discussion than others.

Of further interest are **chains** of links that form over long branches of conversation trees; if we include sentiment and agreement values in links, then patterns of those values in link chains may allow us to categorize conversation types. For example, if a branch of a conversation is dominated by chains indicating agreement and positive sentiment, then that branch can be summarized as an agreement with the point made in the sentence in the initial post.

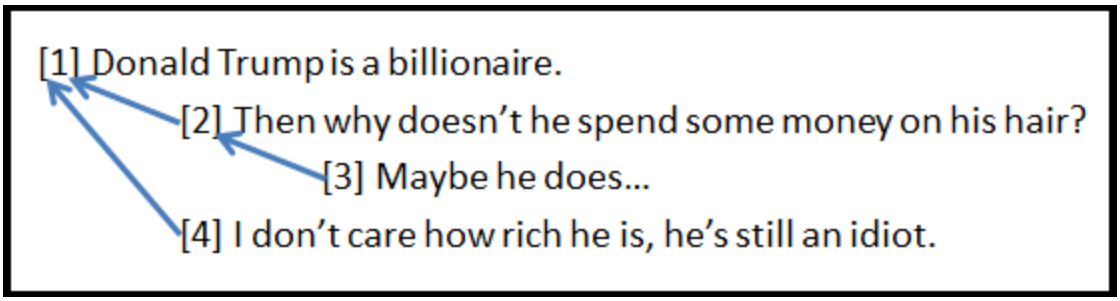

Figure 1. An example of an asynchronous conversation, arranged into the tree representation used by many websites. Sentences 2 and 4 are linked to sentence 1, and sentence 3 is linked to sentence 2.

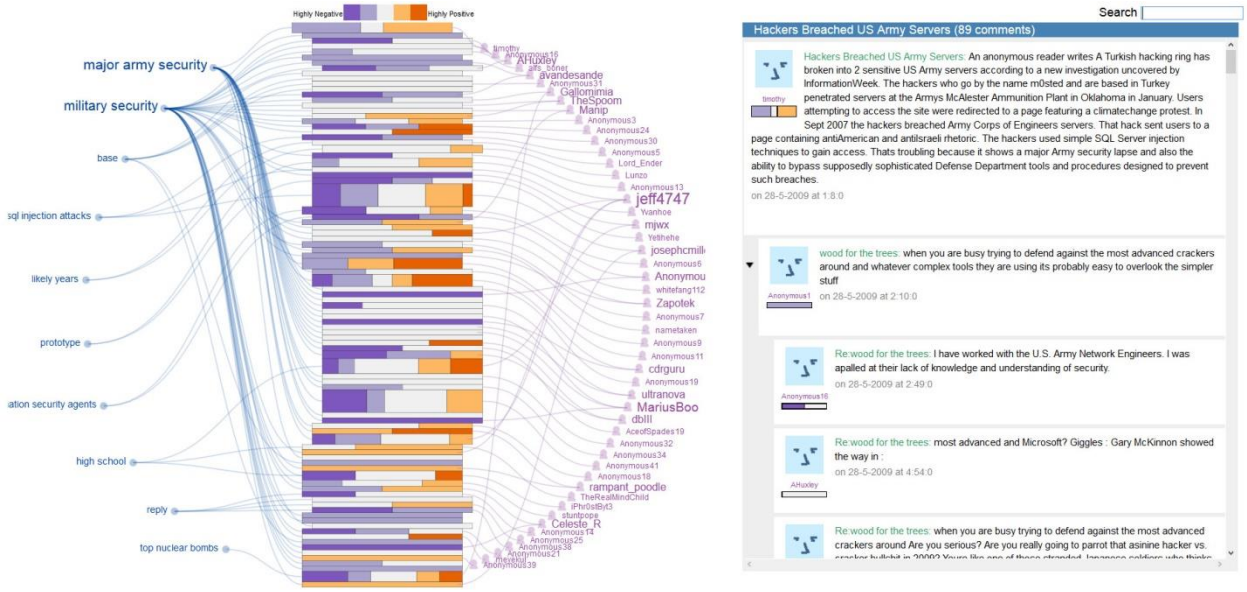

Figure 2. The ConVis system for visualizing asynchronous conversations. The central structure is composed of glyphs representing posts.

1 RELATED WORK

This work extends the functionality of ConVis [1][2], a state-of-the-art tool for visualizing asynchronous conversations that encodes conversation structure, topics, post sentiment, and user activity (See Figure 2). The central feature of ConVis is the set of rectangular glyphs used to represent conversation structure. Each rectangular glyph represents a post; the height of the rectangle encodes the number of sentences in the post, and the children of a given post are positioned below and slightly to the right of it in imitation of the thread structure used by sites that host asynchronous conversations. The distribution of the colours in each rectangle reflects the distribution of sentences in the post with the corresponding sentiment values. Another particularly relevant feature of ConVis is its faceted text view, which allows users to see the text of the posts in the glyph structure. The visualization presented here is intended to extend ConVis to also encode links and allow users to explore and annotate them.

The OnForumS project [3] is an endeavour to combine summarization, argument mining and sentiment mining into a single shared task, of which a portion is creating algorithms to find links between sentences. My exploratory research into this task prompted the idea for the visualization presented here; and OnForumS is the source of the gold standard dataset used in this work.

To my knowledge, there has not yet been any work done in visualizing rhetorical relationships between sentences. The bulk of text-related visualizations seems to have as their scope an entire document or collection of documents, and the data they encode tend to be term frequencies [4] or topics [5].

The visualization idiom used is based on the arc diagram, which dates back to the 1960s as a candidate for graph visualization to minimize edge crossings [6]. In an arc diagram, the nodes are placed on a line, and edges between them are represented by semicircles above and below the line. The ordering of sentences as part of the structure of asynchronous conversations lends itself to this idiom, though the visualization presented here is slightly different in some respects (see Section 4).

2 DATA

The dataset used in this work consists of nine articles, each contained in a single text file, from The Guardian's news website. A tenth file was not used due to time constraints and a formatting issue. Each article is followed by a number of comments, with hundreds of sentences in total for each conversation; and the posts are arranged into the tree structure described earlier using XML tags. Additionally, the sentences are indexed in order from top to bottom as arranged in the tree. These raw files are converted to JSON format for use by ConVis, retaining the structure of the conversation.

A set of dozens to hundreds of links is appended to the end of each file in the dataset. These links are the result of annotations by humans and are intended to serve as a gold standard for research into linking algorithms. Each link contains four attributes applicable to this work:

- **com_sen**: the index of the sentence that is referring to a sentence in an ancestor post. Since this sentence cannot be in an initial article, it must be in one of the comments; hence it is also called the **comment sentence** of the link.
- **art_sen**: the index of the sentence in the ancestor post to which the link refers. Since this sentence may be in an initial article, it is also called the **article sentence** of the link; however, it may be in any ancestor post to that of com_sen.
- **argument**: the agreement value assigned to the link; it takes one of the values {"in_favour", "impartial", "against"}. For example, if a link's argument value is "in_favour", then that link's com_sen agrees with the point made by its art_sen.
- **sentiment**: the sentiment value assigned to the link; it takes one of the values {"positive", "neutral", "negative"}. For example, if a link's sentiment value is "negative", then that link's com_sen is expressing a negative sentiment about the point made by its art_sen. Note that there is a subtle difference between this use of sentiment and the sentiment of a sentence on its own.

In addition to the links' existing attributes, a derived attribute that is useful in this work is the relative position of a given sentence within its post; like com_sen and art_sen, this is an ordered attribute whose value is an integer index. Both argument and sentiment are ordered, diverging attributes.

3 TASKS

I consulted with Giuseppe Carenini and Enamul Hoque, the researchers at UBC responsible for the design and implementation of ConVis. Combining the results of these discussions with my own ideas, I settled on three tasks that the initial offering of this work should facilitate. The first task is to improve the supplied gold standard through revision, with annotators as the target users. The second task is to observe the effect of posts or sentences of interest on descendant branches of the conversation; and the third task is to explore possible link chains related to sentences of interest.

3.1 Task 1: Improving the Gold Standard

A gold standard is intended to be an example of correct human performance in carrying out an NLP task, so that the output of candidate NLP algorithms can be measured against it as an ideal. However, during my exploratory research into the linking task mentioned earlier, I found that a number of the links suffered from various errors, including the following:

- Multiple copies of the same link.
- Multiple links between the same two sentences, but with different attributes. For example, there is a pair of links between the same sentences, but one link has a "positive" sentiment and the other a "negative" sentiment.
- Links where one of the sentences is a single word or a URL. While this is not always an error, the sentence "You?" is unlikely to be reliably linked to one particular sentence over any others.
- Links where one of the sentences consists entirely of punctuation. While "?" does imply confusion, it is also unlikely to be reliably linked to any one sentence.
- Links between unrelated sentences.
- Links with incorrect sentiment and/or argument values. For example, in a conversation about the British government privatizing Royal Mail, one link has the sentence "We need a revolution!" referring to the scandal with the sentiment value "neutral" and the argument value "in_favour".
- Links between sentences in different branches of the conversation tree. While such links may exist due to forum user error, the presence of the other listed errors suggests that some of these links may also be incorrect.
- Links pointing "downwards". The most likely cause of this issue is accidental switching of art_sen and com_sen values, since it is highly unlikely that a forum user would enter a reply above the content being referenced.

In addition, there are issues with the conversation data itself, the most problematic of which is a number of cases where two or more sentences are indexed as a single sentence.

These findings are not the result of an exhaustive assessment of the gold standard; rather, they were found as a result of debugging sessions and sanity checks during the course of the research. Over time it became increasingly clear that a revision of the supplied gold standard would be worthwhile. However, manually assessing nearly two thousand links is a daunting task, made even more so by the fact that the links are appended at the end of each file with sentence indices instead of text in each link. It was evident that a tool would be needed to facilitate the revision, allowing annotators to view linked sentences together and either modify the corresponding link's attributes as needed or delete the link entirely. In addition, since some of the concerns listed involve the positions of the sentences within the conversation tree, a visualization that could provide that context would be appropriate.

In addition to enabling the revision of the existing gold standard, such a tool would also allow researchers to obtain more gold standard data by revising the output of candidate linking algorithms applied to other asynchronous conversations. This process would serve two purposes: expanding the gold standard and evaluating candidate algorithms through analysis of the revisions made.

This task abstracts readily to that of annotation, where some attributes (argument and sentiment) of an object are modified and saved, or where the object is deleted from the dataset. In addition, part of this task is to locate other attributes of the object (com_sen and art_sen) within the overall structure of the dataset.

3.2 Task 2: Observing the Effect of a Post or Sentence

A journalist's success depends at least partially on sparking discussion; and the type of discussion desired, whether thoughtful discussion or hostile debate, varies from writer to writer. It would be useful, then, to see what kinds of discussion arose from different parts of a given article, and which parts of that article sparked the most discussion.

This task abstracts to that of browsing; the user does not know the number or nature of links related to a given post or sentence, but he should be able to tell where to look in order to find them.

3.3 Task 3: Exploring Link Chains

It is not yet certain whether link chains will prove to be significant in NLP analysis; however, the possibility of using them to categorize conversation types is intriguing, so it might prove worthwhile to provide a means to explore an asynchronous conversation in order to find and examine link chains.

This task abstracts to that of exploring in the case where the goal is simply to find any link chains; but it can also be a browsing task if the goal is to find chains that are related to a specific post.

4 SOLUTION

The goal to design the visualization presented in this work as an extension of ConVis, rather than as a standalone visualization, places constraints on the idioms and encoding methods available. However, given the fact that asynchronous conversations naturally employ a tree structure, at least some of those constraints are inherent in any tasks involving their visualization. In addition, while there are many ways to visualize trees, the glyph structure with indentations employed by ConVis is reminiscent of the arrangement used by websites such as Reddit, so many users will be familiar with it. Therefore, if ConVis can be extended to facilitate the desired tasks, then to do so is a reasonable design decision.

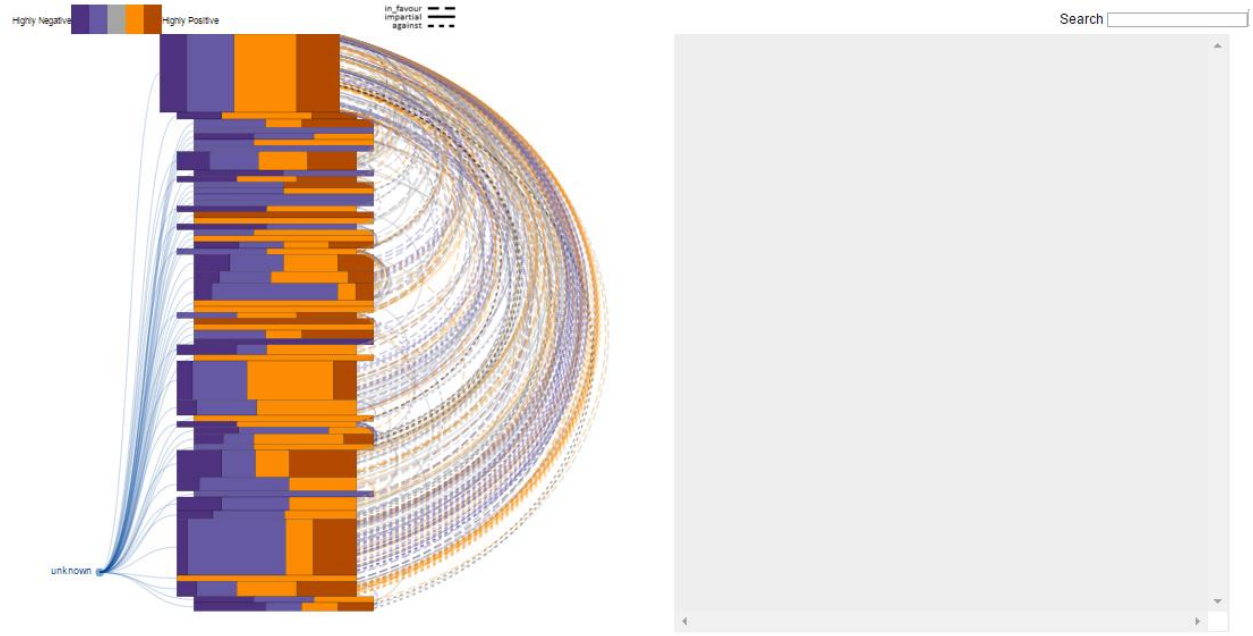

Figure 3. The ConVis extension for visualizing links between sentences in its default state.

The ConVis glyph structure can be viewed as a rough approximation to a vertically straight arrangement of nodes; therefore, this work employs it as the spine of a modified arc diagram (see Figure 3). The links are then encoded as curved line marks that connect the glyphs corresponding to the posts containing the relevant sentences. The visualization does not directly encode com_sen and art_sen; rather, it uses the relative positions of the sentences within their respective posts, and it encodes a relative position via the vertical position of the contact point of the line mark and the appropriate glyph. This design decision—spreading out the links over the side of a glyph rather than attaching them all at a single point—reduces visual clutter in areas with many links and allows users to judge roughly where in a given post a linked sentence will be found.

Since the endpoints of the line marks are not guaranteed to be at the same horizontal position, cubic Bezier curves are used instead of semicircles. Due to links being directed upwards by definition, techniques such as tapered line marks are not required; otherwise, the visual clutter in densely-linked regions would be even worse.
With com_sen and art_sen encoded via a derived attribute, it remains to encode the sentiment and argument values. To maintain consistency with existing ConVis encoding, sentiment is encoded using the same colour scheme as for the glyphs. Unfortunately, the existing ConVis colour scheme has too little contrast with the white background for the line marks to be consistently discernible; therefore, I chose a darker version of that colour scheme for both the glyphs and line marks.

Since bivariate colormaps for two diverging variables can be difficult to read [7], and since any encoding that would modify the width of a line mark would exacerbate the problem of visual clutter in some regions, the visualization encodes argument using the stroke pattern of the line mark. The "impartial" value is assigned the solid line pattern; while that value is thus more prominently displayed than the others, the inequality is at least roughly symmetrical for the diverging values.

Highlighting and filtering line mark selections is done chiefly through animated transitions of line opacity, with selected marks drawn with higher opacity values than the others. In some cases, line width is also increased for selected marks; changes to improve consistency across use cases are slated for future work.

The visualization also employs the ConVis faceted text view in order to display sentences in selected posts, link annotation controls, and/or details of selected link chains. In addition, there is some linked highlighting/filtering functionality between the text view and the line marks to enable users to make meaningful filtering choices. A more detailed description of these relationships is given as part of the use cases in Section 6.

A summary of the solution described here is found in Table 1.

5 IMPLEMENTATION

ConVis is implemented chiefly using PHP, JavaScript, D3 and JQuery. The extension to ConVis presented in this work is implemented using JavaScript, D3 and JQuery.

The drawing and highlighting of the line marks is handled by D3 built-in functions. The HTML content of the faceted text view is generated by custom JavaScript functions. Most of the functions deciding which links to display or highlight were implemented in JavaScript as part of this work as well.

While there is significant room for refactoring and optimization in the implementation of this visualization (see Section 7), one guiding principle during its design was to minimize computation during user interaction. For example, to determine which link trees to display in the chain-building view (see

Section 6.3), a naïve approach would be to check all the links at each step to determine which links are related to the specified sentences. To improve efficiency during chain-building interactions, two adjacency lists are built when the visualization is loaded: one where the specified sentence index is the link's com_sen, and one where the index is its art_sen. When a sentence is added to or removed from the chain, the trees from each end of the chain can be traversed and displayed without repeated searches.

Idiom	**Modified Arc Diagram** (using ConVis glyphs as a spine)
What: Data	Indexed sentences, grouped into article/comments **(tree)** Links with attributes: - Indices of linked sentences **(ordered)** - Sentiment **(ordered, diverging)** - Argument **(ordered, diverging)**
What: Derived	Relative positions of sentences in article/comments **(ordered)**
How: Encode	Curved line marks connecting relative positions of linked sentences in their respective article/comments - Relative positions in article/comments: vertical positions on corresponding ConVis glyph - Sentiment: line color - Argument: line stroke pattern
How: Facet	Use ConVis comment view to display sentences related to selected links
How: Manipulate	Highlight selections; use animated transitions between different highlighted states
How: Reduce	Filter links displayed based on selected article/comment or sentence(s)
Why: Tasks	Annotate: Revise or delete existing links Browse: View number and types of links from article/comment/sentence of interest Explore: Find and examine chains of links Discover Features: Examine attribute patterns in chains of links
Scale	Hundreds of sentences in the article and comments; hundreds of links between sentences

Table 1. Summary of the presented visualization as per Munzner's framework [7].

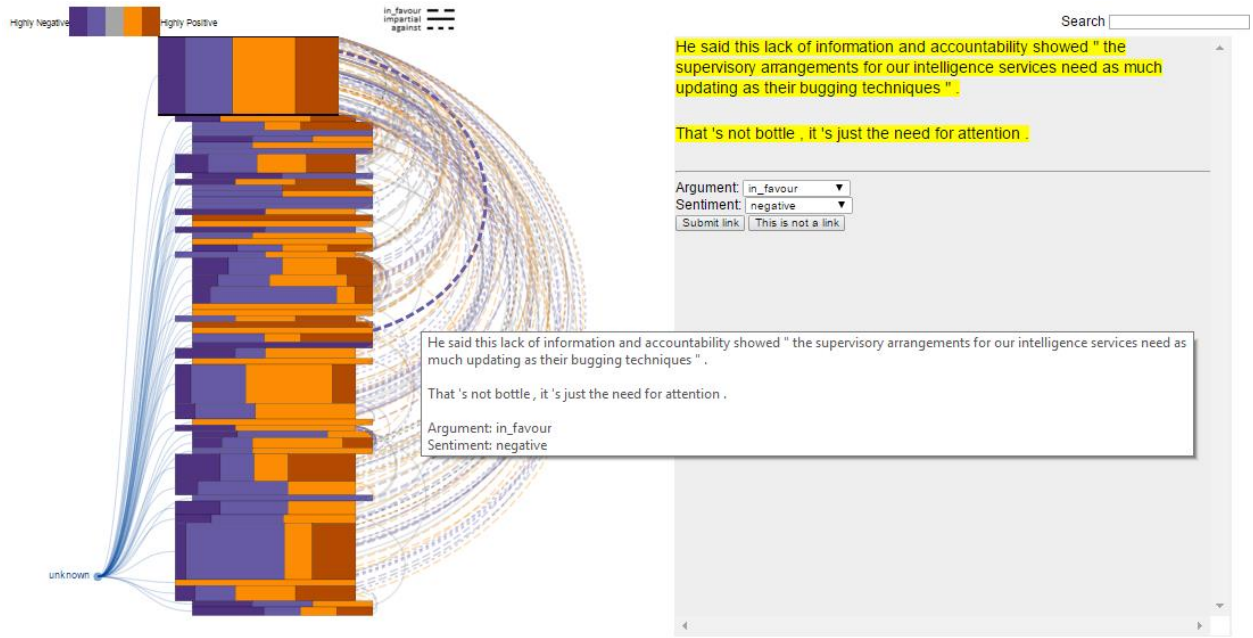

Figure 4. Hovering over a link highlights it and causes a tooltip to be displayed with details related to the link. Clicking on a link highlights it permanently and loads the annotation controls into the comment view.

6 RESULTS

The annotate task (Section 3.1) uses a different set of interactions than the browse (Section 3.2) and explore (Section 3.3) tasks, so use cases are described for each.

6.1 Use Case 1: Deleting an Incorrect Link

An annotator opens the visualization. As a single annotator would not be expected to revise hundreds of links in a single sitting due to concerns of fatigue, she would only see a small subset of the total links for the file. Since her task is to verify, revise or delete all of the links assigned to her, the order in which she handles the links is irrelevant. She moves the mouse over one of the links; it becomes thicker and more opaque, and a tooltip appears containing the text of the linked sentences as well as the argument and sentiment values. She decides to work with that link first and clicks it (see Figure 4).

Once clicked, the link remains highlighted, even when the mouse moves away. The text view changes; it now displays the link's article sentence as well as the entire comment containing its comment sentence. Below the text, controls are displayed that allow the annotator to change the argument or sentiment values; in addition, she may click one of the buttons to submit the link (with revised values if modified) or delete it from the dataset. She looks at the visualization in the centre and notices that the linked sentences are in different branches of the conversation. Since the sentences don't seem strongly related even without that additional context, she decides to delete the link. In either case (submission or deletion), once the action is taken, the link disappears from the visualization. This feature reduces the need for a progress bar; the annotator knows that once all the links are gone, her task is done.

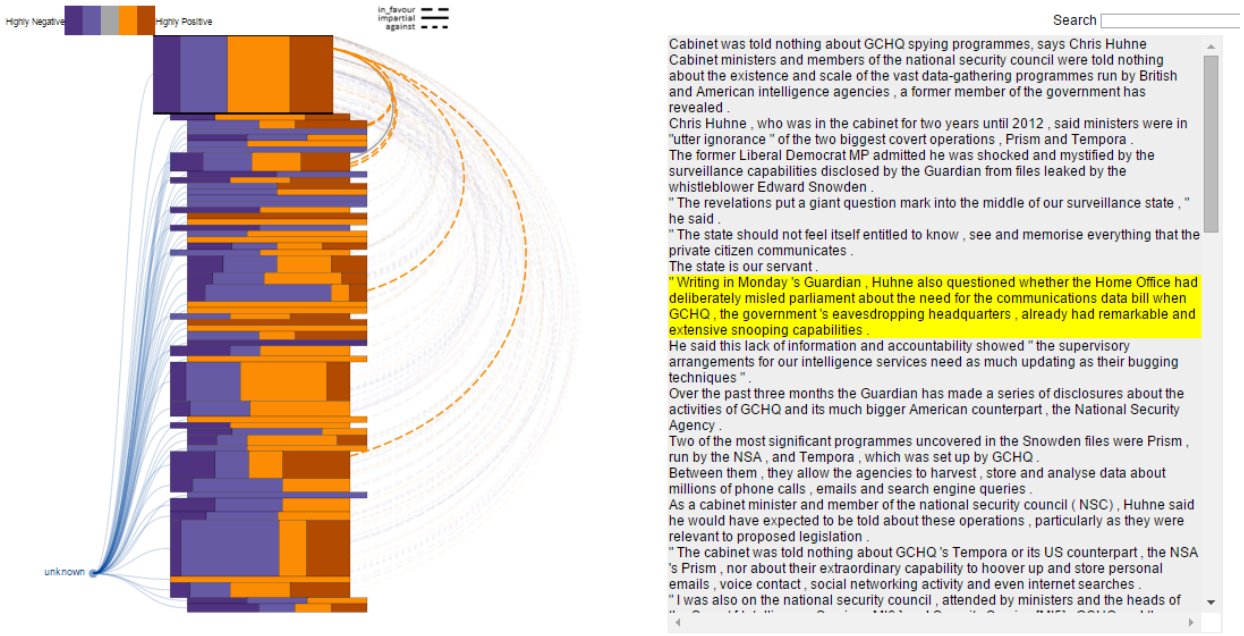

Figure 5. Clicking a post glyph will cause only the links connected to it to be displayed and the post text to appear in the comment view. The displayed links may be further filtered by hovering the mouse over a desired sentence.

6.2 Use Case 2: Browsing an Article for Link-Heavy Sentences

A journalist is interested in public response to an article he wrote recently. In particular, he's trying to improve his writing, and he wants to know which parts of the article sparked the most debate so he can do something similar for his next article. Hovering over links one at a time isn't useful to him; there's too

much clutter at the glyph representing the article. Rather than clicking links, he clicks the article glyph itself.

The visualization now only displays the links related to the article, and the text view now displays a list of the sentences in the article. There are still too many links displayed to make out anything particularly useful, so the journalist moves his mouse over one of the sentences in the text view (see Figure 5). As he moves his mouse from one sentence to another, he sees that the new sentence is highlighted; also, most of the links become very faint, leaving only the links related to the highlighted sentence at full opacity. As he moves from sentence to sentence, he notes which sentences have large numbers of references and whether the references were generally positive or negative.

He pauses on one sentence and notes that there is a large number of links referring to it with negative sentiment, starting from a particular comment. He decides to look at that comment next, to see what was written there, what other parts of the article might have been referenced in it, and how other people responded to it; there may be something there he can use next time.

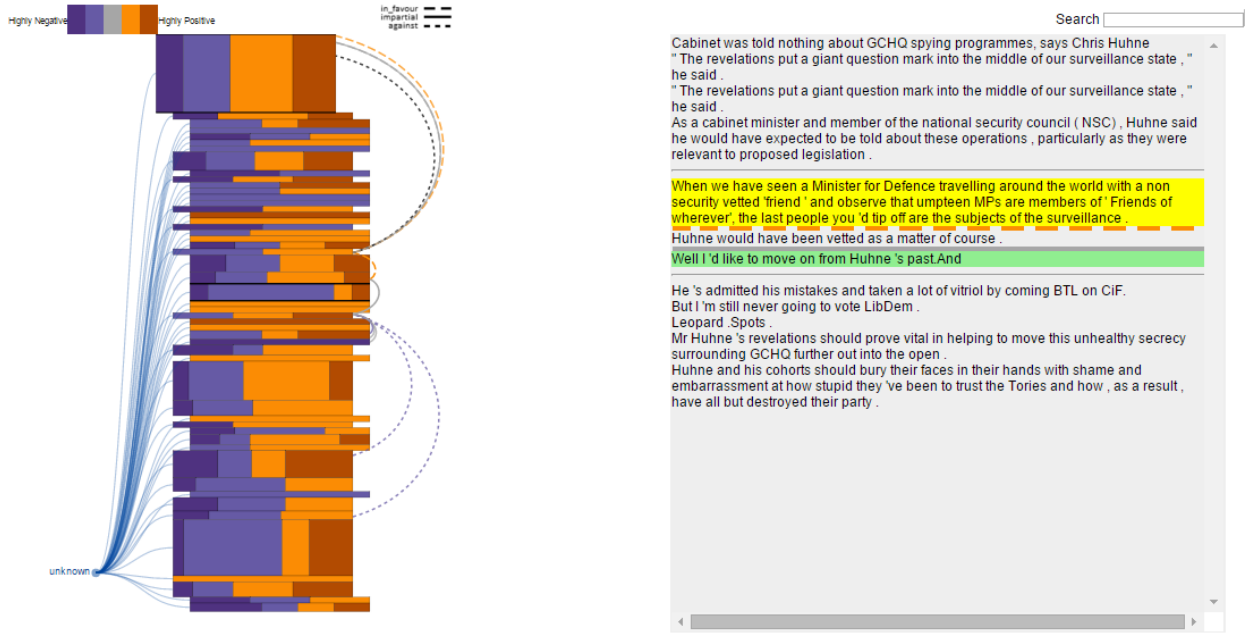

Figure 6. The chain-building view. The current chain is bounded by the sentences at its head (yellow) and tail (green). Each pair of sentences in the chain has a representation of their link displayed between them; the colour and stroke patterns match the line marks in the main visualization. All possible chains that include the current chain are displayed in the main visualization; the result is a tree of links rooted at each end of the chain. Candidate sentences to expand the chain are displayed above and below the current chain sentences.

6.3 Use Case 3: Building a Link Chain

An NLP researcher is intrigued by the idea of link chain characterization, so she opens the visualization with a conversation consisting of several long branches. She follows a similar procedure to that in Section 6.2, but she chooses a long comment in the middle of the longest branch. As she hovers over the sentences in that post, she notices that one sentence has a number of links from farther down the branch as well as some going back up the branch; this may be a good start for a chain. She clicks the sentence.

The text view becomes a chain-building view (see Figure 6); the sentence is highlighted and separated from the rest of the sentences in the view. These other sentences are no longer from the same comment as her chosen sentence, but represent the other ends of every possible link directly connected it. Above the

chosen sentence are the sentences to which it directly refers, sorted in order of position in the overall conversation; below are all the sentences directly referring to her chosen sentence, similarly sorted. To the left of the chain-building text view, every possible chain that includes her chosen sentence is displayed, appearing as two trees branching above and below the chosen comment's glyph.

The researcher moves her mouse over the different sentences above and below her chosen sentence; as each sentence is highlighted, the trees are filtered to show only those chains that also pass through the newly highlighted sentence. She notes that one sentence seems to connect to a longer chain than the others. She clicks that sentence, and it is added to the chain. The link between the two sentences in the chain is now displayed in the text area between them; the colour and stroke pattern match that of the corresponding curve linking the glyphs. The trees of links have been filtered and are now rooted at the corresponding ends of the chain being built. The researcher repeats the procedure, adding sentences to the chain from above and below, observing the trees of links being iteratively filtered until there are no more options. Now the text view has a list of all the sentences in the chain, with link representations between each pair. If she wishes to backtrack and examine a different chain, she may do so by clicking the head or tail of the chain, as desired.

7 DISCUSSION AND FUTURE WORK

While the visualization presented here has potential as an extension of ConVis, there are a number of areas in which it can be improved.

One of the more important areas for improvement is code quality and optimization, since improvements there will likely simplify improvements in other areas. While some attention was paid to optimizing code for low-overhead interactions (see Section 5), a thorough code review is warranted. In hindsight, a proper low-level design phase was never conducted, partially due to unforeseen delays; it would have been more efficient and effective than the "design-on-the-fly" paradigm employed for some features of the visualization.

There are some inconsistencies in user interaction and system response across the use cases. For the annotation task, for example, users interact with the line marks and the controls in the text view; while for the other tasks, users interact with the post glyphs and the sentences in the text view. Given that these tasks are intended for two different kinds of users, that may not end up being a design flaw; however, additional feedback on those design choices is warranted. Highlighting is also handled differently in the annotation task. An attempt to make highlighting consistent across all use cases resulted in unwanted behaviour interaction, so the change was reverted. Once the code is refactored, it is likely another attempt will be successful, which should reduce concerns about visual clutter somewhat.

While filtering and selection functionality is available, there is currently no aggregation or filtering based on link attributes. I plan to combine the sentiment and argument legends into a scented widget that provides counts of each permutation of attributes currently active, and that allows the user to choose whether to display each one.
Currently, if two linked sentences are close to each other, the majority of the link between them can be occluded by a protruding post glyph. I plan to revisit the equations determining the Bezier guidepoint locations in an attempt to remedy the situation for at least some such cases.

The visualization does not currently fit on a standard laptop screen, though this may also be a current issue with ConVis. Many of the variables governing visualization size are hardcoded, and so refactoring the code to handle different screen sizes may be worthwhile.

The text area looks quite crude at the moment; this is due to the priority of functionality over appearance. I plan to take time to address this concern as well.

Since the line marks encoding the links are quite thin, it can be difficult to interact with them or even to tell them apart. I plan to look into adding some sort of zoom functionality to increase the ease-of-use of the system for annotators.

Currently, the presented visualization replaces the commenter view currently implemented in ConVis. If ConVis is to be extended, a way to switch between views will need to be implemented. Ideally, the areas on both sides of the post glyphs should be able to house any of the visualizations ConVis offers; and being able to choose different combinations of them should mean allowing extra functionality depending on which combination is chosen. Additional discussion on this matter is warranted.

There is currently only one implemented way to explore and examine link chains. If link chains prove useful in NLP research, additional chain-related functions should be added. For example, a researcher may want to examine all possible chains between two selected posts, one at a time, using a scroll wheel; or she may want to filter chains by length or attribute pattern. There are many possible chain-related interactions that may be worth consideration.
Due to time constraints and the priority of vis-related functionality, the ability to store revised link attributes and save them to file is not yet implemented.

When an annotator is evaluating a link, the article sentence is displayed without its surrounding sentences providing context, but the comment sentence's entire post is shown. While this prevents the annotator having to scroll through a long article to find the right sentence, it doesn't prevent her having to scroll through a long comment. I plan to implement sliders that will allow the annotator to decide how much context is to be displayed for both linked sentences.

Finally, after at least some of the items discussed in this section are completed, user studies should be designed and performed to help determine the effectiveness of this visualization.

Once we have a fully operational and effective extension to ConVis, a high priority is to revise and expand the available gold standard data for sentence links. The current dataset does not contain very long branches, and most of its links connect directly to the articles rather than to other ancestor posts. Additional data, properly annotated, may allow us to better examine the potential of link chains as useful tools in NLP research.

8 CONCLUSION

I have presented an extension to ConVis that allows users to perform varied tasks related to annotating and exploring links and link chains in asynchronous conversations. I have discussed and justified design choices made in its development, and I have presented use cases for the tasks the visualization was designed to facilitate. While there are many ways in which this work can be improved, it has potential to be a useful extension to a state-of-the-art tool.

ACKNOWLEDGMENTS

I wish to thank Giuseppe Carenini, Enamul Hoque, Tamara Munzner and Vaden Masrani for their design feedback, technical instruction and/or assistance with debugging.

REFERENCES

[1] Hoque, Enamul, and Giuseppe Carenini. "ConVis: A visual text analytic system for exploring blog conversations." Computer Graphics Forum. Vol. 33. No. 3. 2014.

[2] Hoque, Enamul, and Giuseppe Carenini. "Convisit: Interactive topic modeling for exploring asynchronous online conversations." Proceedings of the 20th International Conference on Intelligent User Interfaces. ACM, 2015.

[3] Kabadjov, Mijail, et al. "OnForumS: The Shared Task on Online Forum Summarisation at MultiLing'15." Proceedings of the 7th Forum for Information Retrieval Evaluation. ACM, 2015.

[4] Rohrer, Randall M., David S. Ebert, and John L. Sibert. "The shape of Shakespeare: visualizing text using implicit surfaces." Information Visualization, 1998. Proceedings. IEEE Symposium on. IEEE, 1998.

[5] Liu, Shixia, et al. "Tiara: Interactive, topic-based visual text summarization and analysis." ACM Transactions on Intelligent Systems and Technology (TIST) 3.2 (2012): 25.

[6] Saaty, Thomas L. "The minimum number of intersections in complete graphs." Proceedings of the National Academy of Sciences of the United States of America 52.3 (1964): 688.

[7] Munzner, Tamara. Visualization Analysis and Design. CRC Press, 2014.