Why Visualize Chat?

Chat, particularly in the form of instant messaging, is becoming an increasingly frequently used channel for communication in corporate environments. A number of researchers have studied the use of instant messaging (IM) in the workplace, and have found that it is used for everything from short questions and coordinating availability [5] to complex tasks [4].

While the “instant” in instant messaging is often taken to refer both to the speed of interaction and the ephemeral nature of the messages, messages – although created on-the-fly – sometimes persist. First, we have data from admittedly unsystematic surveys that suggest a majority of power users of one IBM chat client save some or all of their IM transcripts. Second, in a study of a broadcast messaging system used to initiate group chats at IBM [6], 29% percent of users who had broadcast a question reported saving the complete transcript of the resulting chat, and another 9% reported saving parts of the transcript.

Third, also within IBM, ethnographic studies of a small group of consultants indicate that saving instant messaging transcripts is not uncommon. Halverson [3] reports an in depth study of a single consultant who saves all his chat transcripts. He not just saves chats, but actively organizes and searches them. Saving, accessing and mining chats is a critical component of
his job – providing expert level help to other consultants involved in high profile mission critical engagements. We believe that such behavior, while exceptional now, is a harbinger of the future and suggest that, given the increasingly important role played by chat in business settings, that explorations of how to save, organize, browse and search chat archives would provide much of value.

**Visualizing Messaging Histories**
A key issue in making persistent chat more useful is to help users access the content of their logs. The naïve approach of just providing a search function would scratch only the surface of the wealth of information that can be discovered in such a collection because it neglects exposing structure within conversations, (temporal) relationships between conversations and the social aspects of such a collection.

In its simplest form a conversation browser would help users re-find information they know is contained in a previous conversation where it is difficult to define a query. A typical scenario would be "to find a URL I know Ted mentioned in a chat roughly 2 weeks ago". Another might be "to find the email of that guy in group G who used to work with Steve – Steve mentioned him in a conversation last month and gave me his email address". These scenarios illustrate several typical aspects of such message retrieval:

- Users start with vague or incomplete information.
- Users might be looking for data embedded in the conversation stream, such as URLs, email addresses, phone numbers and the like.
- Queries are often based not on the information sought for, but on another aspect of the conversation (context) the user remembers.
- Searches often hinge on a person. Users remember that they got the information from X. This is an important point because it means that even if we don’t find the information in the transcript we can get in touch with that person and try to retrieve the information from them.
- Typically we have a coarse idea of a time range where that information was given. The further back in time we go the less accurate this time component tends to get.

While a lot of work has been done in email systems to support re-finding information and understanding threads of communication (e.g., [2]), much less effort has been spent on tools to understand persistent messaging. There are a couple of unusual aspects of instant messaging that are good starting points to visually distinguish messages in a social visualization:

- Messages can vary a lot in character. Some are very short (e.g. “going to lunch now”), others are lengthy. Some people write in short spurts, others tend to formulate a longer thought before hitting send.
- Conversations vary in turn-taking behavior, and in extent of multi-threadedness.
- Chats can be more or less synchronous. There may be long pauses while people are away.
from their desks, such that the exchange develops an email-like character.

- Conversations across different time zones might span multiple days and still be considered one conversation.

- Chats can be among more than 2 participants, some of which join later and not all of whom may have a complete transcript [1].

- Messaging logs are kept locally on a machine. However, users may switch machines during an extended chat and end up with incomplete logs on each machine. Users, however, might well remember this as a single conversation.

These and related issues are not only challenges but also opportunities, because each of them is a relevant feature of the conversation that can be used to better characterize it. A social visualization that lets users explore their conversation logs would hinge on exposing such features and users might dynamically show and hide display of various features in the visualization. For instance features might differentiate between very short and lengthy conversations, mark conversations that contain email addresses or attachments, mark conversations that occurred at unusual times (conference trip, couldn’t sleep), conversations that span days, contain many long gaps or conversations that involve several people. Incidentally these are exactly the kind of features used in the vague queries in the scenarios above.

**A Design Sketch: Chat Lens**

In this section we describe a visualization designed to take advantage of the features we’ve discussed. Our approach builds on previous work on a time-line based visualization called the ContextLens [1]. A context lens, in essence, is a representation of a time-line split into segments. For each segment the visualization indicates whether a specified feature is present in the segment.

Figure 1 shows a sketch of the Chat Lens. To the left is a temporally ordered representation of the chat: time goes from top to bottom, the left columns indicate utterances by speakers, and the right columns indicate the presence of specific features. To the right is a fly-out window that shows the actual chat (both the highlighted utterance and its context). The Chat Lens has been configured to show the presence of three out of four participants, and the presence of URLs. Columns

![Fig 1: Design sketch for a Chat Lens indicating speaker and embedded URLs. (The column showing Jack’s utterances has been hidden).](image)
representing other features – such as phone numbers, pictures, attachments – could be added using the “+” indicator. Features can also include names in a corporate address book or any kind of relevant keyword such as a competitor’s name, project names, etc.

By just glancing at the Chat Lens it is easy to see the presence and rough distribution of features within the conversation. Imagine the scenario that I remember that somebody mentioned a URL during the conversation with Jack, but I don’t know who it was. By turning on a feature that shows activity by Tom and that recognizes URLs, the Chat Lens reveals which parts of the conversation to look at. Hovering over the indicated segment lets me examine the transcript of these relevant parts and minimizes the amount of chat transcript I actually have to read. This ability to provide additional detail on demand is one of the reasons we are using this visualization. It effectively allows us to provide a minimal visualization but still gives easy access to additional context information when needed.

While this approach to visualizing chat seems useful, there is a potential problem: chats are often very long, and in such cases an utterance by utterance visualization would not scale well. A solution to this problem is to use temporal aggregation to produce a more compact representation, that can be revealed hierarchically. That is, rather than each row of the visualization representing an utterance, it instead represents a time period such as a minute or an hour. The representation of utterances and features work as before, except that the representation shows either simple binary indicators of presence or absence, or alternatively the frequency of occurrence using gray scale, etc. Hovering over a segment of a hierarchical Chat Lens again reveals a pop-up window showing a Chat Lens for that time span, and so on.

Figure 2 shows an example of such a hierarchical Chat Lens. In this particular case each of the higher-level segments shows one hour of chat. The segments show the distribution of features, as gray scale, aggregated into segments of 10 minutes. In the 2nd level popup the chat transcript for each hour is shown as 2-minute segments which again give easy access to the actual transcript. Note that the pop-up window indicates the presence of an attached file. This could easily be shown as a feature in the ChatLens if the user wishes. By
adjusting aggregation and segment size it is easily possible to navigate a larger time span, such as a few days or a week, find those parts of the conversation the user is interested in based on context and quickly access those parts of conversation.

The idea of Chat Lenses (and hierarchical Chat Lenses) is quite powerful. In particular, the notion of features is quite general. Features might be extended to include work context (such whether the user was in her office or traveling), which machine(s) she used, and whether a particular application was used during the chat. Chat specific features might also depict the social context of the conversation and indicate that I had a chat with somebody else in parallel, that I took a phone call, whether the conversation occurred while in a meeting, while working from home or in a different time zone at a conference or a number of other features that could be detected from the work context or the content of the conversation (such as the always popular swear-word and flame detector). Features might also be aggregated to show, for instance, conversations with a member of a group instead of with a specific person or the presence of word from a keyword list.

**Summary**

We have learnt to accept email systems as a focal point of our work day and as repository were we share and keep information. We see that persistent chat has the potential to become another highly relevant repository for information and artifacts about our work, only in a less formal way. To be really useful, we need to devise systems that let us easily organize conversation logs and find information again in them. As chat is less formal than email it is important to support very vague queries into the corpus of communication by exposing structure in the corpus and within the individual communications.

**References**


