Extending Word Highlighting in Multiparticipant Chat

David C. Uthus
NRC/NRL Postdoctoral Fellow
Washington, DC 20375
`david.uthus.ctr@nrl.navy.mil`

David W. Aha
Navy Center for Applied Research in AI
Naval Research Laboratory (Code 5514)
Washington, DC 20375
`david.aha@nrl.navy.mil`

Abstract

We describe initial work on extensions to word highlighting for multiparticipant chat to aid users in finding messages of interest, especially during times of high traffic in chat rooms. We have annotated a corpus of chat messages from a technical chat domain (Ubuntu's technical support), indicating whether they are related to Ubuntu's new desktop environment *Unity*. We also created an unsupervised learning algorithm, in which relations are represented with a graph, and applied this to find words related to Unity so they can be highlighted in new, unseen chat messages. On the task of finding relevant messages, our approach outperformed two baseline approaches that are similar to current state-of-the-art word highlighting methods in chat clients.

Introduction

This work addresses a problem in the US Navy where chat rooms, which are being used for command-and-control communications, are contributing to information overload (Catanzaro et al. 2006). More generally, chat rooms can be difficult to follow when the tempo is fast and multiple conversations are taking place simultaneously. Word highlighting can help a user to focus on messages of interest. Many chat clients allow a user to enter *user-specified words* (USWs) and will then subsequently highlight these words in all the messages. Unfortunately, word highlighting has its limits: it can fail when words are misspelled, abbreviated, or replaced with an acronym. It can also fail when similar words are used instead of those requested by the user. We are addressing the problem of how to find additional messages of interest to bring to a user's attention given their set of USWs.

We propose an approach to extend word highlighting by giving chat clients the ability to highlight additional words that are similar to the USWs. We focus on highlighting additional, related words since highlighting individual words makes it possible to know why a message may be relevant when compared to highlighting a whole message. We created an unsupervised learning approach called *Graph-based Word Relation Highlighter* (GWRH) that finds relations between words using a graph representation, and then uses these relations to find words related to those in a USW list. When applied, it returns a ranked list of words, ordered by the strength of their relations to those the user requested.

We begin this paper by first describing the problem, the related work, and the corpus we have created for this investigation. This is followed by a description of GWRH and then our empirical studies that show GWRH outperforms two baseline approaches that are similar to current state-of-the-art highlighting techniques in chat clients. We finish with conclusions and possible future extensions of our work.

Problem Description

Given a set of unlabeled messages U, a test set of labeled messages L, and a set of USWs W, the goal is to learn a model such that, when given a word $w \in W$, the model will return a set of related words R. With this set of related words, for each message $l \in L$, mark as positive (in terms of message to be highlighted) any message containing a word $r \in R \cup \{w\}$. Messages not to be highlighted are left with a negative label. The overall goal is to find a set of words that maximizes the cover of positive labels in L while minimizing the coverage of negative labels in L.

There is the issue of whether it is better to have more false positives or false negatives. For this research, false positives are more desirable since the user can decide themself whether a highlighted message is relevant – with false negatives, the user may miss the message altogether.

Related Work

The state-of-the-art in highlighting for chat clients (e.g., popular clients such as mIRC[1], XChat[2], Pidgin[3], and Konversation[4]) is rather simplistic. These clients allow a user to enter their USWs (some allowing regular expressions), and will then highlight those words for them.

[1]`http://www.mirc.com`
[2]`http://xchat.org`
[3]`http://www.pidgin.im`
[4]`http://konversation.kde.org`

```
[05:40] <daddy> is there a way to change
11.04 interface back to 10.10
[05:40] <DrFrankenstein> daddy:  launching
programs from a drop down menu instead of
the screen with the icons?
[05:40] <Soupermanito> yes, log out and
choose clasic interface at the log in menu
```

Figure 1: Example chat conversation where the users are discussing about Unity through inference (Unity is the 11.04 default interface).

One related research area concerns the topic detection and tracking of chat messages, specifically applications of unsupervised learners (Bingham, Kabán, and Girolami 2003; Kolenda, Hansen, and Larsen 2001). The difference between these approaches and what we are investigating here is that these approaches are topic-focused – they group words only within a set of topics whereas we instead find relations independent of topics. For example, Bingham, Kabán, and Girolami required the number of topics to be estimated, which limits the granularity of the topic groups. Kolenda, Hansen, and Larsen also had a set number of topics and a reject group, where messages not fitting in the set topics would be put in this latter group. For the problem we are investigating, these approaches would be useful when a user is interested in a word that can be found in one of these topics, but would fail should a user be interested in a word not grouped under one of these topics. Additionally, for these approaches, the strength of a word is in relation to a topic, while we are interested in strengths of relationships between words.

Another area of related work are approaches for finding messages of relevance, particularly from a military perspective (Berube et al. 2007; Budlong, Walter, and Yilmazel 2009; Dela Rosa and Ellen 2009). All three of these approaches required prior knowledge to determine what is important: Berube et al. used regular expressions and entity classes; Budlong, Walter, and Yilmazel used a rule-based algorithm and a statistical analysis approach using maximum entropy; and Dela Rosa and Ellen used supervised learning algorithms. New types of important messages would not be detectable without new regular expressions, rules, or labeled data. This differs from our goal of an approach that can adapt as new types of messages are created.

Corpus

We have created a corpus for testing using a subset of chat logs from the Ubuntu Chat Corpus (Uthus and Aha 2013). This corpus is composed of two parts: a training set of seven days of unlabeled chat messages, beginning from Ubuntu 11.04's release date (28 April to 4 May 2012) and a test set of hand-annotated messages taken from one day (5 May 2012), which took us three weeks to annotate. The annotated messages have a binary label – whether they are related to a specific set of USWs. As a starting point, we have a single USW, "Unity[5]," since this was a hot topic following the release of Ubuntu 11.04. The training set consists of 81,848 messages and the test set consists of 8675 messages, with 468 of these labeled as positive.

```
[01:24] <bible-boy> anything else like unity
that i can use on 11.04
[01:24] <bible-boy> cause i like unity
[01:24] <TomRone> bible-boy, install fluxbox
desktop environment with synaptic and give
that a shot or lxde perhaps.  you use the
login manager to choose which environment to
use
[01:25] <lapion> bible-boy, why can't you
use unity ?
[01:25] <bible-boy> where can i download
that because i have it installed on a pc
without internet.  Right now im running
Ubutnu off a live cd
[01:25] <bible-boy> oh i cant use it because
i only have 512mb memory
```

Figure 2: Example chat conversation showing how a conversation can shift in topic. It began with Unity, but branched to also discuss a different desktop environment. Red highlighted messages are messages about Unity.

Labeling this data requires determining whether a message is related to a particular topic. This is difficult due to the conversational (threaded) nature of the chat messages and the inference that is usually drawn from knowledgeable chat users about a specific topic. An example of this is shown in Figure 1, where user `daddy` asks a question about Unity, referring to it as the "11.04 interface." For the corpus, we labeled only those messages that concern topics related to Unity – this then results in some conversation threads to only be partially labeled as positive due to the changing of topics within a conversation. An example of this can be seen in Figure 2, where a part of the conversation shifts from Unity to an alternate desktop environment.

Graph-based Word Relation Highlighter

Figure 3 shows GWRH's general framework. It begins by applying an unsupervised learning algorithm on U, which results in graph G. This graph is then used to find R based on the words in W. Given a message to be checked m, GWRH will then highlight all words in m from $W \cup R$, returning m'.

GWRH creates an undirected graph where each node represents a word from the corpus and the edges are the weights (strength of relations) between words. More specifically, the edges connect all pairs of words that appear together in a message and the weights represent the number of times they appear together.

UpdateGraph

Procedure UpdateGraph describes how new messages are applied to the graph to update edge weights. For each mes-

[5]Unity was a new desktop environment introduced by Ubuntu: `http://unity.ubuntu.com`

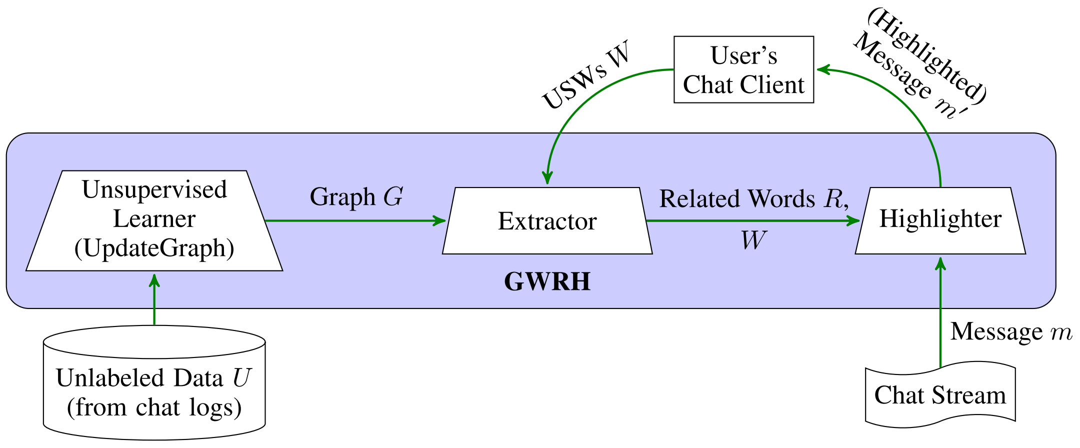

Figure 3: GWRH's general framework and how it interacts with a user's chat client.

sage $u \in U$, it will change the weights of G based on the words found in u. For normalization, all text is changed to lower case. For preprocessing, all name mentions are removed when the first word of a message matches the name of an author seen in the past 100 messages. This limit was chosen arbitrarily. Should a message u contain words not seen before, the algorithm will add new vertices to the graph.

```
Procedure UpdateGraph(message $u$, graph $G$)
  $u' \leftarrow$ normalize_and_preprocess($u$)
  $u^T \leftarrow$ tokenize_and_remove_duplicates($u'$)
  for each $t \in u^T$ do
      if vertex $v_t \notin G$ then $G \leftarrow G \cup \{v_t\}$
  $P \leftarrow$ all_pairs($u^T$)
  for each pair $p$ in $P$ do
      if edge $e_p \in G$ then
          $e_p \leftarrow e_p + 1$  // update edge weight
      else
          $G \leftarrow G \cup \{e_p\}$
          $e_p \leftarrow 1$
  return $G$
```

Extractor

Procedure Extractor describes how the set of related words R (ranked by the strength of the relations) is extracted from G given the set of USWs W. For each $w \in W$ and each neighbor n of w, Extractor first calculates two ratios: one is the ratio of the edge between n, w and the summed weights (ϵ_w) of edges connecting to w, and the other is the ratio of the same edge and summed weights (ϵ_n) of edges connecting to n. These ratios (along with the associated neighbor's identity) are stored in two lists, L_w and L_n.

When calculating these weights, we only consider edges that have at least a minimal weight limit of λ. This helps reduce possible noise in the graph, especially of accidental mentions between unrelated words.

```
Procedure Extractor(words $W$, graph $G$, filter $\lambda$)
  // $W$ is the set of USWs
  $R, L_w, L_n \leftarrow \emptyset$        // reset ranked lists
  for each word $w \in W$ do
      $N \leftarrow$ neighbors($G, w$)
      $\epsilon_w \leftarrow \sum_{n \in N} e_{w,n} : e_{w,n} > \lambda$
      for each neighbor $n \in N$ do
          $\epsilon_n \leftarrow \sum_{v \in \text{neighbors}(G,n)} e_{n,v} : e_{n,v} > \lambda$
          $L_w \leftarrow L_w \cup \{(e_{w,n}/\epsilon_w, n)\}$
          $L_n \leftarrow L_n \cup \{(e_{w,n}/\epsilon_n, n)\}$
      sort($L_w$)              // sort $w$'s ratios
      sort($L_n$)    // sort neighbor's ratios
      for each neighbor $n \in N$ do
          $n\_avg\_rank \leftarrow \frac{1}{2} \cdot$ (rank($L_w, n$)+rank($L_n, n$))
          $R \leftarrow R \cup \{n\_avg\_rank\}$
  return sort_by_average_rank($R$)
```

After creating L_w and L_n, Extractor sorts them to compute the rankings for each neighbor. Ties are broken by order of insertion into the graph. It will then compute the average rank, and at the end, return the set of related words sorted by their average rank.

We create two rankings to filter out common words that appear frequently (e.g., stop words and words that are common technical terms but not generally considered stop words in traditional texts). Ranking by L_w will give us a list of words that appear most frequently with w, while ranking by L_n will give us a list of words that consider w to be of relevance to them. The latter list then helps filter out the list of the former, since common words will have w ranked low on their lists.

To illustrate how these rankings are calculated, we refer to Figure 4. Suppose the value of ϵ for the neighboring nodes are already calculated and $\epsilon_{\mathtt{w}} = 50$. For each neighboring node, Extractor will calculate two weight ra-

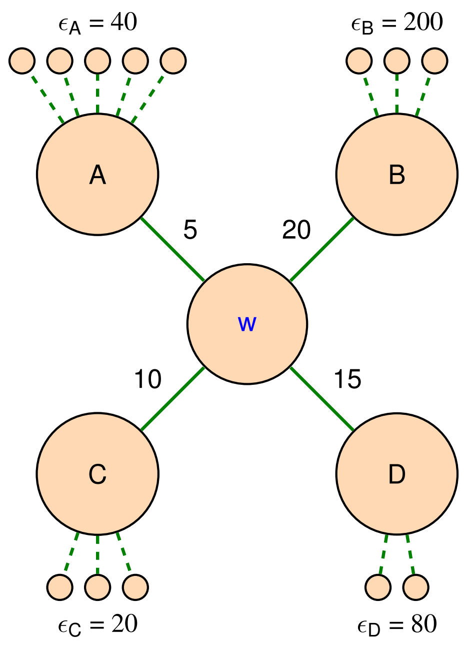

Figure 4: An example graph to use for Procedure Extractor.

tios, one with ϵ_w and one with $\epsilon_{i\in\{A,B,C,D\}}$. This will result in $L_w = \{\frac{5}{50}, \frac{20}{50}, \frac{10}{50}, \frac{15}{50}\}$ and $L_n = \{\frac{5}{40}, \frac{20}{200}, \frac{10}{20}, \frac{15}{80}\}$ with respect to the four neighbors. When sorted and ranked, A would have ranks 4 & 3, B would have ranks 1 & 4, C would have ranks 3 & 1, and D would have ranks 2 & 2, all with respect to L_w and L_n. When these ranks are averaged and sorted, the neighboring nodes would then be in order of {C, D, B, A}.

Extractor is executed only when the user submits a new set of W – it does not update R every time G is updated with a new message m. We chose this design since it is not desirable to change the related set of words without the user knowing, nor do we want to bother the user every time G is updated.

Highlighter

In regular use, the highlighter method would take in R and W. It will make use of any words in R that the user would want highlighted (e.g., allowing a user to pick which additional words to highlight). For each new message m that it receives from the chat stream, it will check to see if there are any words in m that match a word in $R \cup W$, and if so, highlight these words. It will then pass the message to the user's chat client.

For the empirical studies described next, after extracting R, GWRH will iterate through L and mark as positive all messages that contain at least one word $r \in R \cup W$. With respect to R, it will be restricted so only the top ranked words will be used, with more detail described in the subsequent section. It will then evaluate the messages to check which are correctly marked as positive and negative.

Highlighting Algorithm	Precision	Recall	F_2 score
baseline_s	0.71	0.52	0.55
baseline_t	1.0	0.5	0.56
$\text{GWRH}_{\lambda=9}$	0.49	0.65	0.61

Table 1: Results of our baseline approaches and GWRH when $\lambda = 9$.

Empirical Study

We conducted a set of experiments to evaluate GWRH by comparing it to two baseline approaches that are similar to the current state-of-the-art capabilities of most IRC clients for word highlighting. One baseline approach checks to see if w is a substring of a message $l \in L$ (called *baseline*$_s$); the other baseline approach checks if w matches a tokenized word in l (called *baseline*$_t$). We hypothesize that GWRH can achieve better recall and F_2 scores than our baseline approaches.

For our experiments, we used the corpus described earlier in this paper. All messages were normalized by changing all characters to lower case. They were then tokenized using the NLTK tokenizer (Loper and Bird 2002) (with exception for baseline_s). GWRH was applied such that it used the top ten ranked words in R and "unity" (the use of top ten was chosen arbitrarily, though in future work this can be tested as a parameter). For metrics, we use standard precision and recall, along with F_2, which puts greater emphasis on recall than precision.

Table 1 shows the results of running these baseline approaches and GWRH with $\lambda = 9$. Comparing the two baselines, searching for substrings decreases precision when compared to matching string tokens due to the word "unity" being a substring of common words like "community", while only gaining a small increase in recall. These approaches only achieved recalls of 50% and 52% respectively, showing that there are many messages about Unity that do not explicitly mention it. GWRH, compared to the two baselines, achieves higher recall and F_2 scores, which provides informal support for our hypothesis.

Examining how λ effects GWRH, we ran a series of tests with increasing values of λ. Figure 5 shows the results of these tests. As can be seen, increasing λ initially increases recall and precision, then has a smaller impact as it grows further in value. This can be attributed to λ reducing the noise from rare mentions of unrelated words.

The top ten ranked terms when $\lambda = 9$ are *gnome, classic, 2d, 3d, compiz, interface, launcher, plugin, bar,* and *desktop*. Unsurprisingly, many of these words share strong relations to Unity, either changing to alternative or past desktops (gnome, classic), synonyms for a desktop environment (interface), components of Unity (launcher, bar, desktop, plugin), software Unity is built on (compiz) or concern issues people had running Unity on older hardware (2d, 3d). More importantly, we can see that in these top ten words, common English words are removed despite their frequent appearance. This is desirable since it allows GWRH to return

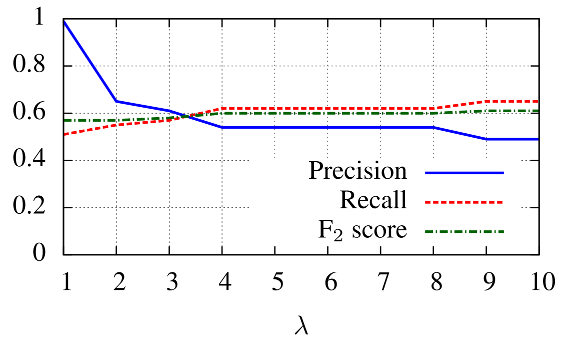

Figure 5: Results for different values of λ (parameter to filter out noise).

words that have stronger relevance to the USWs and less relevance to words not in the USW list.

Conclusion

We have presented a new approach to extend word highlighting in multiparticipant chat. GWRH can find word relations using a graph-based unsupervised learning algorithm. Our results show that, for one corpus (concerning technical support), it outperforms two baseline approaches that are similar to current state-of-the-art chat client capabilities.

While GWRH increases recall for our task, there are some types of messages that it would not easily be able to find. One example is messages that use pronouns to refer to Unity. Another is messages that use a misspelling that forms another correctly-spelled word (e.g., "unify" instead of "unity").

In our future research, we will first try to extend the corpus, allowing for formal validation of our hypothesis. As it took three weeks to validate the 8675 messages in our test set, we would then expect it to take many months to annotate enough messages to allow for cross-validation. To alleviate this, we are investigating how to leverage crowd sourcing for annotating chat messages. In addition to annotating a longer series of messages in time, we will also obtain annotations for additional topics (i.e., other than Unity).

One possible extension for GWRH is to consider conversation threads. We could use these to train the model. When looking at Figure 3, the unlabeled data could be disentangled using a thread disentanglement method (Elsner and Charniak 2010; Wang and Oard 2009) prior to being passed to the unsupervised learner. Instead of changing weights by examining only a single message in a vacuum, we could change the weights by examining the message in context to its given conversation. As a part of thread disentanglement, we will also consider disambiguating pronouns. As discussed before, finding related words will not always help when Unity is referred to by a pronoun.

Another interest is extending GWRH to be a lifelong learner, as we want it to learn new terminology as they are introduced. Most of our approach is suitable to lifelong learning, though adjustments need to be made to prevent the graph from growing too large. The graph created for these experiments included 35,281 vertices and 1,699,164 edges. Of these edges, 65.8% have a weight of 1, meaning the two vertices of the edge have only been seen once together. It could be possible then to prune such edges over time, and even remove vertices once they no longer have any edges. Additionally, the approach needs to be modified so it can "forget" relations since terminology changes with time.

Finally, these ideas of extending word highlighting will be evaluated through human subject studies in a simulated Navy environment. As mentioned earlier, our work addresses a problem in the US Navy of chat and information overload. We hope that our results will show that extending word highlighting can assist Navy watchstanders with finding messages of interest in a fast-paced environment.

Acknowledgments

Thanks to NRL for funding this research. David Uthus performed this work while an NRC postdoctoral fellow located at the Naval Research Laboratory. The views and opinions contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of NRL or the DoD.

References

Berube, C. D.; Hitzeman, J. M.; Holland, R. J.; Anapol, R. L.; and Moore, S. R. 2007. Supporting chat exploitation in DoD enterprises. In *Proceedings of the International Command and Control Research and Technology Symposium*. CCRP.

Bingham, E.; Kabán, A.; and Girolami, M. 2003. Topic identification in dynamical text by complexity pursuit. *Neural Processing Letters* 17:69–83.

Budlong, E. R.; Walter, S. M.; and Yilmazel, O. 2009. Recognizing connotative meaning in military chat communications. In *Proceedings of Evolutionary and Bio-Inspired Computation: Theory and Applications III*. SPIE.

Catanzaro, J. M.; Risser, M. R.; Gwynne, J. W.; and Manes, D. I. 2006. Military situation awareness: Facilitating critical event detection in chat. In *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting*, volume 50, 560–564. Human Factors and Ergonomics Society.

Dela Rosa, K., and Ellen, J. 2009. Text classification methodologies applied to micro-text in military chat. In *Proceedings of the International Conference on Machine Learning and Applications*, 710–714. IEEE Computer Society.

Elsner, M., and Charniak, E. 2010. Disentangling chat. *Computational Linguistics* 36(3):389–409.

Kolenda, T.; Hansen, L. K.; and Larsen, J. 2001. Signal detection using ICA: Application to chat room topic spotting. In *Proceedings of the Third International Conference on Independent Component Analysis and Blind Source Separation*, 540–545.

Loper, E., and Bird, S. 2002. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 63–70. Philadelphia, PA: Association for Computational Linguistics.

Uthus, D. C., and Aha, D. W. 2013. The Ubuntu Chat Corpus for multiparticipant chat analysis. In *Proceedings of the AAAI Spring Symposium on Analyzing Microtext*. AAAI.

Wang, L., and Oard, D. W. 2009. Context-based message expansion for disentanglement of interleaved text conversations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 200–208. ACL.