2007

Lexical and Discourse Analysis of Online Chat Dialog

Forsyth, Eric N.

http://hdl.handle.net/10945/39587

Lexical and Discourse Analysis of Online Chat Dialog

Eric N. Forsyth and Craig H. Martell
Department of Computer Science, Naval Postgraduate School
enforsyt@nps.edu, cmartell@nps.edu

Abstract

One of the ultimate goals of natural language processing (NLP) systems is understanding the meaning of what is being transmitted, irrespective of the medium (e.g., written versus spoken) or the form (e.g., static documents versus dynamic dialogues). Although much work has been done in traditional language domains such as speech and static written text, little has yet been done in the newer communication domains enabled by the Internet, e.g., online chat and instant messaging. This is in part due to the fact that there are no annotated chat corpora available to the broader research community. The purpose of this research is to build a chat corpus, tagged with lexical (token part-of-speech labels), syntactic (post parse tree), and discourse (post classification) information. Such a corpus can then be used to develop more complex, statistical-based NLP applications that perform tasks such as author profiling, entity identification, and social network analysis.

1. Introduction

In 2006, Jane Lin [1] collected 475,000+ posts made by 3200+ users from five different age-oriented chat rooms at an Internet chat site. The chat rooms were not limited to a specific topic, i.e. were open to discussion of any topic. Lin's goal was to automatically determine the age and gender of the poster based on their chat "style". The features she captured were surface details of the post, namely, average number of words per post, vocabulary breadth, use of emoticons, and punctuation usage. Lin relied on the user's profile information to establish the "truth" of each user's age and gender.

The data Lin captured has enormous potential, and as such has formed the foundation of an ongoing research effort at the Naval Postgraduate School's Autonomous Systems Laboratory. Specifically, the goals related to this effort include the following: 1) preserve the online chat dialog in an XML-based corpus to aid in future accessibility to the data; 2) annotate the chat corpus with lexical, syntactic, and discourse information; and 3) use this annotated corpus to develop, train and test higher-level NLP applications.

There are numerous NLP applications that could benefit from an annotated chat corpus. For example, law enforcement and intelligence analysts could use author profiling and entity identification applications to help detect predatory or terrorist activities on the Internet. On the other side of the spectrum, legitimate chat use could be enhanced by applications that automatically identify and group the multiple threads of conversation that often occur within chat.

2. Building the Corpus

The Python programming language was the primary tool we used to build the corpus. Within Python, we used Lundh's ElementTree module [2] to create, edit, store, and retrieve the XML documents that comprised the corpus. We also used Schemenauer's back-propagation neural network Python class [3] for our automated post classification effort. In addition, Loper and Bird's Natural Language Toolkit Lite (NLTK-Lite) Python modules [4] formed the basis for our automated lexical analysis. Finally, we used an XML parser for subsequent corpus editing and validation.

One of the challenging aspects we faced in developing the corpus was sanitizing it to protect user privacy. If the corpus is to be made available to the larger research community, this must be accomplished. It was straightforward to replace the user's screen name in both the session logs as well as the user profile with a mask, for example, "killerBlonde51" with "10-19-30sUser112." However, more often than not, users were referred to by variations of their screen names in other users' posts. For example, other users would refer to "killerBlonde51" as "killer", "Blondie", "kb51", etc. Although regular expressions can assist in the masking task, ultimately 100% masking requires hand verifying that the appropriate masks have been applied in every post. To date, complete masking has been accomplished on 3,507 (~700 posts/chat room) of the 475,000+ posts.

It should be noted that although masking is essential to ensure privacy, it results in a loss of information. For example, the way to which users are referred often conveys additional information, for example, familiarity and emotion; this information is lost in the masking process. In addition, it was observed that a user's screen name would become a topic of conversation independent from the original user; again, the origin of this conversation thread is lost in the masking process.

3. Discourse Analysis: Post Classification

A great deal of research has been performed regarding discourse analysis of spoken language. Stolcke, et al [5] developed over 40 tags associated with different dialog acts used in conversational speech. Certainly, a fundamental reason why online chat is similar to spoken conversational speech is that a conversation is taking place. In addition, fillers like "you know", "really" as well as interjections like "hey" and "awww" occur both in speech and online chat. However, with chat, multiple topics are being discussed by multiple people simultaneously, and people don't always "wait their turn" when posting. Finally, the stops and restarts associated with spoken dialog do not seem to occur in chat.

Obviously, chat is also very similar to written text. However, chat participants often spell words phonetically, e.g. "dontcha" for "don't you". In addition, they make extensive use of emoticons and abbreviations, e.g. ":-)" and "LOL" (Laughing Out Loud). Finally, due to the nature of the medium, words are frequently misspelled.

Recognizing these distinctions, Wu, et al [6], used subsets of previous dialog act tags along with chat-specific tags to automatically classify 3,129 chat posts over Internet Relay Chat channels into 1 of 15 categories using Transformation-Based Error Driven learning.

As an initial annotation attempt for our online chat corpus, we classified the 3,507 user-sanitized posts mentioned earlier using Wu's 15 post categories, and investigated two different machine learning algorithms to automatically classify the posts. Wu's classification categories as well as an example of each taken from our corpus are shown below.

Table 1. Post classification examples

Classification	Example
Accept	yeah it does, they all do
Bye	night ya'all.
Clarify	i meant to write the word may.....
Continuer	and thought I'd share
Emotion	lol
Emphasis	Ok I'm gonna put it up ONE MORE TIME 10-19-30sUser37
Greet	hiya 10-19-40sUser43 hug
No Answer	no I had a roomate who did though
Other	0
Reject	u r not on meds
Statement	Yay...democrats have taken the house!
System	JOIN
Wh-Question	11-08-20sUser70 why do you feel that way?
Yes Answer	why yes I do 10-19-40sUser24, lol
Yes/No Question	cant we all just get along

These examples highlight the complexity of the task at hand. First, we should note that posts were classified into only one of the 15 categories. At times, more than one category might apply. In addition, the "Wh-Question" example does not start with a "wh" token, while the "Yes Answer" does start with a "wh" token. Also, notice that the "Yes/No Question" does not include a question mark. Finally, the "Statement" example contains a token that conveys an emotion ("yay"). Taken together, these examples highlight the fact that more than just simple regular expression matching is required to classify these posts accurately.

The initial post classification task was assisted by simple regular expression matching, followed by hand correction of each post. Of these posts, various, randomly-selected subsets were used for training (3007 posts total) and testing (500 posts total). The overall frequencies of the post classes in our sanitized corpus are shown below. Note that the highest occurring category of posts was "Statement", with more than double the next highest classification category.

Table 2. Post classification frequencies

Class	Count	Percent
Statement	1210	34.50%
System	597	17.02%
Greet	470	13.40%
Emotion	404	11.52%
Wh-Question	187	5.33%
Yes/No Question	183	5.22%
Continuer	122	3.48%
Accept	86	2.45%
Reject	75	2.14%
Bye	55	1.57%
Yes Answer	41	1.17%
No Answer	33	0.94%
Emphasis	17	0.48%
Other	15	0.43%
Clarify	12	0.34%

The machine learning algorithms we used require a set of features on which to base their automated classification. The definition of the set of features used is shown below, with a brief discussion following.

1. Number of posts ago the poster last posted (normalized by max session length).
2. Number of posts ago that a post led with a yes/no question or included a “?” pattern (normalized by max session length).
3. Number of posts in the future that contain a yes or no pattern (normalized by max session length).
4. Number of posts ago that a post led with a greet pattern (normalized by max session length).
5. Number of posts in the future that led with a greet pattern (normalized by max session length).
6. Number of posts ago that a post led with a bye pattern (normalized by max session length).
7. Number of posts in the future that led with a bye pattern (normalized by max session length).
8. Number of posts ago that a post was a JOIN (normalized by max session length).
9. Number of posts in the future that a post is a PART (normalized by max session length).
10. Total number of users currently logged on (normalized by max users in the session).
11. Total number of tokens in post (normalized by max length post in train/test set).
12. First token in post contains hello or variants
13. First token in post contains goodbye or variants.
14. First token in post contains wh-question start such as who, what, where, etc.
15. First token in post contains yes/no-question start such as is, are, does, etc.
16. First token in post contains conjunction start such as and, but, or, etc.
17. Number of tokens in the post containing one or more “?” (normalized by maximum number of ? found in a single post in train/test set).
18. Number of tokens in the post containing one or more “!” (normalized by max number of “!” found a single post in train/test set).
19. Number of tokens in the post containing yes or variants (normalized by max number of yes variants found in a single post in train/test set).
20. Number of tokens in the post containing no or variants (normalized by max number of no variants found in a single post in train/test set).
21. Number of tokens in the post containing emotion variants such as lol, ;-), etc (normalized by max number of emotions found in a single post in train/test set).
22. Number of token(s) in the post in all caps, e.g. JOIN (normalized by max number of tokens in caps found in a single post in train/test set).

Features 1-9 of a post are based on the posts surrounding it, specifically, the distance to posts with particular features, with the rationale that surrounding posts should give a hint to the nature of the post itself. For example, “Continuer” posts should be more likely to follow fairly closely to when the user last posted, and “Yes/No Answers” should follow fairly closely to posts with yes/no question characteristics. Feature 10 (current number of users logged on) was selected because it might help normalize the distances associated with Features 1 through 9 (with the rationale that more users currently logged on might increase those distances). Feature 11 is based on the post itself, with the rationale that the number of tokens will give a good initial hint at what the post is, e.g., longer posts being perhaps “Statements”, and shorter posts being perhaps “Emotions” or “Yes/No Answers”. Finally, Features 12-22 are also based on the post itself, but are looking for specific patterns which should give a clue on the nature of the post. For example, “Greet” posts should contain a token like “hello”, while “Yes/No Questions” and “Wh-Questions” might contain “?” as a token.

3.1. Post classification learning algorithm #1: Back-propagation neural network

The initial machine learning method we investigated to classify posts was a back-propagation neural network. Specifically, it employed the following sigmoid activation function

$$f\left(x\right) = \arctan\left(x\right)$$

In addition, it consisted of input nodes, output nodes, and a single hidden layer of nodes, as well as learning rate and momentum factors. So, for our model, we had 22 input nodes (the number of features), 15 output nodes (the number of post classes), 14 hidden nodes, a learning rate of 0.05, and no momentum. We did not perform a global optimization on the hidden layer, learning rate, and momentum parameters. Instead, we varied them around set values and selected the configuration that reduced the error the most after twenty iterations on each configuration.

Precision, recall, and f-scores for each of the classes for one instance of a training/test set are shown below. Note that after training, we selected the output vector with the highest firing rate as the post classification of the test data fed into the neural net.

Table 3. Example neural net results

Class	Test Freq	Prec	Recall	FScore
Accept	16	0.417	0.313	0.357
Bye	2	0.667	1.000	0.800
Clarify	5	undef	0.000	undef
Continuer	15	undef	0.000	undef
Emotion	64	0.873	0.750	0.807
Emphasis	3	undef	0.000	undef
Greet	66	0.935	0.879	0.906
nAnswer	4	undef	0.000	undef
Other	3	undef	0.000	undef
Reject	12	0.500	0.250	0.333
Statement	164	0.670	0.915	0.773
System	78	0.975	1.000	0.987
whQuestion	32	0.909	0.625	0.741
yAnswer	8	undef	0.000	undef
ynQuestion	28	0.667	0.857	0.750

Performance of this neural net was comparable to the results obtained by Wu with Transformation-Based Error Driven learning. As with Wu, the neural net does not appear to be able to make a reasonable classification unless a class appears in greater than 3% of the postings. Most of the misclassifications occur in the “Statement” class. We believe the reason for this is the fact that the “Statement” class is the maximum likelihood estimate (MLE) for the labeled data set. In other words, given no other information, the most likely label for a particular post is the Statement class based on the overall frequency of Statements in the data set. In particular, the frequency of Statements is twice that of the next highest category.

3.2. Post classification learning algorithm #2: Naïve Bayes

In addition to the neural network approach, we investigated using the Naïve Bayes machine-learning algorithm to classify posts. By Bayes Rule

$$P\left(C_i \mid f_1 \wedge f_2 \wedge \ldots \wedge f_n\right) = \frac{P\left(f_1 \wedge f_2 \wedge \ldots \wedge f_n \mid C_i\right) P\left(C_i\right)}{P\left(f_1 \wedge f_2 \wedge \ldots \wedge f_n\right)}$$

But by assuming independence among the variables we classify a post according to

$$C = \arg\max_i \left[P\left(f_1 \mid C_i\right) P\left(f_2 \mid C_i\right) \ldots P\left(f_n \mid C_i\right) P\left(C_i\right)\right]$$

As with the neural network, we used the same 22 features as input to the algorithm. To estimate the actual probability distribution represented by our training data, we used “add-one”, or Laplace smoothing (see Mitchell’s discussion of the m-estimate for a fuller account [7]). Precision, recall, and f-scores for each of the classes for one instance of a training/test set using the Naïve Bayes approach are shown below.

Table 4. Example Naïve Bayes results

Class	Test Freq	Prec	Recall	FScore
Accept	13	0.250	0.154	0.190
Bye	6	0.500	0.167	0.250
Clarify	1	undef	0.000	undef
Continuer	13	0.500	0.077	0.133
Emotion	63	0.846	0.524	0.647
Emphasis	4	undef	0.000	undef
Greet	76	0.849	0.816	0.832
nAnswer	5	undef	0.000	undef
Other	4	undef	0.000	undef
Reject	9	0.000	0.000	undef
Statement	170	0.552	0.871	0.676
System	79	0.987	0.987	0.987
whQuestion	25	0.762	0.640	0.696
yAnswer	7	undef	0.000	undef
ynQuestion	25	0.429	0.120	0.188

As can be seen, Naïve Bayes as implemented appears to perform less well than the 22 feature neural network model shown earlier. To formally compare the performance between the two learning approaches, we randomly selected 30 train/test sets for each model, and calculated the mean and standard deviation of their f-scores. Due to time constraints, we limited the number of iterations for the neural network models to 100 for each of the 30 samples. We then performed a hypothesis test on two populations to see if there is a significant difference in the performance between the models. For 95% confidence, we reject the null hypothesis that the means are equal if $|z| > 1.96$. The results are shown below.

Table 5. Learning algorithm FScore comparison

	NN Vector		Bayes Vector		
Class	Mean	Std Dev	Mean	Std Dev	z
Accept	undef	undef	undef	undef	undef
Bye	0.761	0.140	undef	undef	undef
Clarify	undef	undef	undef	undef	undef
Continuer	undef	undef	undef	undef	undef
Emotion	0.802	0.042	0.615	0.061	13.950
Emphasis	undef	undef	undef	undef	undef
Greet	0.890	0.022	0.831	0.026	9.612
nAnswer	undef	undef	undef	undef	undef
Other	undef	undef	undef	undef	undef
Reject	undef	undef	undef	undef	undef
Statement	0.786	0.019	0.681	0.024	18.757
System	0.972	0.020	0.976	0.014	0.959
whQuestion	0.791	0.040	0.576	0.078	13.439
yAnswer	undef	undef	undef	undef	undef
ynQuestion	0.690	0.068	0.360	0.092	15.805

4. Lexical Analysis: Part of Speech Tagging

As dialog act classification forms the basis of discourse analysis, part-of-speech (POS) tagging is a fundamental form of lexical analysis, and is a critical input to higher order NLP tasks such as parsing. As such, we want to build highly accurate POS taggers to automatically annotate our online chat corpus. The ultimate accuracy of POS taggers for a particular domain depends on two aspects: 1) the algorithm used to make the tagging decision; and 2) if statistically-based, the data used to train the tagger.

The basic tagging algorithm we implemented involved training a bigram tagger, backing off to a unigram tagger, backing off to the maximum likelihood estimate tag; we'll subsequently refer to this as our bigram backoff tagger. Working backwards, the maximum likelihood estimate tag is the most common tag within the training set.

$$t_i = \arg\max_t \left[\text{count}(t) \right]$$

A unigram tagger assigns the most common POS tag to a word based on its occurrence in the training data.

$$t_i = \arg\max_t \left[P\left(t_i \mid w_i \right) \right]$$

Finally, a bigram tagger assigns the most common POS tag to a word not only based on the current word, but also the previous word as well as the previous word's POS tag.

$$t_i = \arg\max_t \left[P\left(t_i \mid w_i \wedge t_{i-1} \wedge w_{i-1} \right) \right]$$

Thus, our tagging approach works as follows: The tagger will first attempt to use bigram information from the training set. If no such bigram information exists, it will then back off to unigram information from the training set. If no such unigram information exists, it will finally back off to the MLE tag for the training set.

Several POS-tagged corpora in many languages are available to NLP researchers. The corpora we used to train various versions of our taggers are contained within the Linguistic Data Consortium's Penn Treebank distribution [8]. The first corpus, referred to as Wall Street Journal (WSJ), contains over one million POS-tagged words collected in 1989 from the Dow Jones News Service. The second, referred to as Switchboard, was originally collected in 1990 and contains about 2,400 transcribed, POS-tagged, two-sided telephone conversations among 543 speakers from all areas of the United States. Finally, the third, referred to as Brown, consists of over one million POS-tagged words collected from 15 genres of written text originally published in 1961. All corpora were tagged with the Penn Treebank tag set.

In addition to the aforementioned Penn Treebank corpora, 1,391 POS-tagged posts from our chat corpus were used to train/test various versions of our taggers. The posts (a subset of our 3,507 user-sanitized posts) were initially tagged with a bigram/regular expression tagger trained on Switchboard and Brown and then hand-corrected. In the end, the 1,391 posts provided a total of 6,078 POS-tagged words (tokens). Although the posts were tagged using the Penn Treebank tag set and associated tagging guidelines [9], we had to make several decisions during the process that were unique to the chat domain.

The first class of decisions regarded the tagging of abbreviations such as "LOL" and emoticons such as ":-)" frequently encountered in chat. Since these expressions conveyed emotion, they were treated as individual tokens and tagged as interjections ("UH").

The second class involved words that, although would be considered misspelled by traditional written English standards, were so frequently encountered within the chat domain that they were treated as correctly spelled words and tagged according to the closest corresponding word class. As an example, the token "wont" (when referring to "won't"), if treated as a misspelling, would be tagged as "^MD^RB", with the "^" referring to a misspelling and "MD" and "RB" referring to "modal" and "adverb", respectively. However, since it was so frequently encountered in the chat domain, we tagged it as "MD".

The final class of decisions involved words that were just plain misspelled; in that case, they were tagged with the misspelled version of the tag. As an example, "intersting" (when referring to "interesting") was tagged as "^JJ", a misspelled adjective.

However, before determining what the most accurate bigram backoff tagger for the chat domain was, we first needed a baseline comparison. To do this, we trained and tested a bigram tagger for each of the other domains, using the same amount of data as we had for the chat domain. Since we had 1,391 tagged chat posts, one might be inclined to select training/test sets consisting of 1,391 sentences from the other domains. However, the unit of concern is at the token-, and not sentence-level. Therefore, this would be inappropriate, since Treebank corpora sentences were much longer than chat posts. Since the 1,391 tagged chat posts contained 6,078 tokens, we randomly selected contiguous sections of the Wall Street Journal and Switchboard corpora, each containing at least 6,078 tokens (plus the tokens necessary to complete the last sentence) to serve as source data for those domains. From those selections, we created 30 different training/test sets by randomly removing ~14.4% of the sentence-level units from each domain to serve as test data with the remainder serving as training data. Summary statistics for the corpora selections as well as their associated bigram backoff tagger performance are shown in the table below.

Table 6. Corpora tokens and types example

	Chat	WSJ	Switch
Sentence-Level Units	1391	106	412
Tokens	6078	6107	6079
Types	1477	1891	921
Tokens/Type	4.115	3.230	6.600
Bigram Accuracy (mean)	0.737	0.722	0.802
Bigram Accuracy (std dev)	0.014	0.013	0.015

Again, the purpose of this initial analysis was to determine, when given an equivalent amount of data to train and test from, how the bigram backoff tagger trained and tested on chat compares to similar taggers for the WSJ and Switchboard domains. Clearly, the performance of the chat domain tagger is on par with the other domains. However, notice the trend that as the Tokens/Type figure increases, the accuracy of the tagger also increases. For the WSJ and Switchboard domains, this particular training/test selection is typical when compared to the mean and standard deviations of 30 contiguous samples taken from each domain—see Table 7 below. This makes sense from a qualitative standpoint, since as the number of tokens for a particular type increases, the more data there is available for a statistical tagger to base a tagging decision on. This, however, is not the only measure of a domain's linguistic variety at the lexical level. Certainly, looking at only the types of lemmas is something that could be taken into account when considering lexical variety. Also, the greater the number of POS tags for a particular type, the more difficult it will be for a tagger with a limited context such as the bigram tagger to make the correct tagging decision given a limited amount of data. That being said, it is interesting to note that, based on the tokens/type figure alone, chat is significantly more varied lexically than transcribed speech, being much closer to the WSJ written text domain. More importantly, though, this snapshot, although based on a specific test/training size, provides a level of confidence that state-of-the-art statistical taggers employed on chat should reach similar accuracy rates given similar amounts training data.

Table 7. Corpora tokens/type (~6078 tokens per sample, 30 contiguous samples)

	WSJ	Switch
Mean Tokens/Type	3.221	6.614
Std Dev	0.180	0.308

The question here, of course, is exactly what sort of non-chat data should we use to train our chat tagger on. The following table provides the mean tagging accuracy and associated standard deviations for 30 test sets (200 posts/test set) for five different bigram backoff taggers trained on the following corpora: 1) WSJ; 2) Switchboard; 3) Brown; 4) All three Treebank corpora; and 5) The remaining 1,191 POS-tagged chat posts.

Table 8. Bigram backoff tagger accuracy based on training corpus

	WSJ	Brown	Switch	Treebank	Chat
Mean Accuracy	0.574	0.583	0.621	0.658	0.737
Std Dev	0.019	0.022	0.017	0.017	0.014

Clearly, the bigram backoff taggers trained on chat perform significantly better than the other taggers, even though the Treebank-based taggers were trained on millions of words (compared to thousands of words for the chat taggers). This is not surprising, since chat has a vocabulary quite different from the other domains, to include the extensive use of emoticons and abbreviations which appear nowhere in the Treebank domains. It is interesting to note how taggers trained on Switchboard perform significantly better than those trained on other Treebank domains. This is due in part to the fact that Switchboard contains several interjections used extensively in chat that are simply not found in the other domains, to include “yeah”, “uh-uh”, “hmm”, “Hi”, etc.

Given that training on chat seems to be the best single data source for building a chat POS tagger, can we still take advantage of the vast amount of POS data collected from other domains? To explore this, we modified our chat bigram backoff tagger in the following way. Instead of backing off from chat bigram to chat unigram to finally the chat MLE tag, after not encountering chat unigram information, back off instead to a bigram tagger trained on another domain, followed by the other domain’s unigram tagger, and finally to the chat MLE tag. Below is the mean tagger accuracy for this approach, with the secondary bigram backoff taggers trained on individual Treebank domains as well as all three Treebank domains.

Table 9. Combined bigram backoff tagger performance

	Chat to WSJ	Chat to Brown	Chat to Switch	Chat to Treebank
Mean Accuracy	0.851	0.858	0.855	0.871
Std Dev	0.012	0.012	0.010	0.012

As can be seen, all represent significant improvements in tagger accuracy over the bigram backoff tagger based solely on chat training information. It is interesting to note that the apparent advantage of the Switchboard data disappears when the tagger is first trained on chat. This is because the additional interjection vocabulary is already contained within the chat data itself, and thus the presence of it in Switchboard adds nothing to overall tagger performance. In the end, the 87.1% accuracy for the chat to Treebank bigram backoff tagger is significantly the best tagger of the entire set of taggers investigated. We believe that a slight modification to this relatively simple tagger can still yield accuracy dividends. For example, before making the final back off to the chat MLE, we could incorporate a regular expression trained on the morphology of words, e.g. tagging all sanitized users as proper nouns (NNP, since we know the format for the user sanitization scheme), tagging all words ending in “ing” as gerund verbs (VBG) and all words ending in “ed” as past tense verbs (VBD), etc.

5. Future Work

Our initial efforts in preserving and annotating the online chat corpus appear promising. As such, we have a number of future efforts planned to continue improving automated lexical and discourse annotation performance. With regards to POS tagging, we must first complete the hand tagging of the full 3,507 user sanitized posts (2,116 remaining). With our current bigram backoff tagger approaching 90%, this should be accomplished relatively quickly. In conjunction with this, we need to investigate more sophisticated POS taggers, to include Hidden Markov Model and Brill’s Transformational Based Learning tagging [10] approaches. It is our belief that the additional chat training data and more sophisticated tagging algorithms, when combined with the Treebank data, should yield tagging accuracy performance above 90% range. We also will revisit our decision to tag both emoticons and chat abbreviations as “UH”, since much of its usage in the Switchboard corpus is reserved for speech disfluencies (and thus may have a different distribution than in our chat corpus). We will accomplish this by adding one or more tags to cover emoticon and chat abbreviation usage, and compare subsequent tagger performance with the original "single tag for all interjections" approach.

Improved POS data can then be used in modifying the feature set for the post classification discourse analysis, which currently does not include any POS tag features. Finally, more sophisticated smoothing approaches should improve the performance of the Naïve Bayes-based post classification performance.

6. References

[1] J. Lin, *Automatic Author Profiling of Online Chat Logs*, M.S. Thesis, Naval Postgraduate School, Monterey, 2007.

[2] F. Lundh, http://effbot.org/zone/element-index.htm, Python ElementTree Module, 2007.

[3] N. Schemenauer, http://arctrix.com/nas/python/bpnn.py, Python Back-Propagation Neural Network Class, 2007

[4] E. Loper and S. Bird, "NLTK: The Natural Language Toolkit", *Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, Association for Computational Linguistics Somerset, NJ., 2005, pp 62-69.

[5] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, M. Meteer, and C. Van Ess-Dykema, "Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech". *Computational Linguistics*, 2000.

[6] T. Wu, F.M. Khan, T.A. Fisher, L.A. Shuler, & W.M. Pottenger, "Posting Act Tagging using Transformation-Based Learning" *The Proceedings of the Workshop on Foundations of Data Mining and Discovery*, IEEE International Conference on Data Mining (ICDM'02), December 2002.

[7] T.M. Mitchell, *Machine Learning*, McGraw Hill, Singapore, 1997.

[8] M.P. Marcus, B. Santorini, M. Marcinkiewicz & A. Taylor, *Treebank-3*, Linguistic Data Consortium, Philadelphia, 1999.

[9] B. Santorini, "Part of Speech Tagging Guidelines for the Penn Treebank Project", *Treebank-3*, Linguistic Data Consortium, Philadelphia, 1999.

[10] E. Brill, "A Simple Rule-based Part of Speech Tagger", *Proceedings of the Third Conference on Applied Natural Language Processing*, 1992, pp 152-155.

[11] R. Hwa, "Supervised Grammar Induction using Training Data with Limited Constituent Information", *Proceedings of the 37th conference on Association for Computational Linguistics*, 1999, pp 73-79.