Vote Prediction on Comments in Social Polls

Isaac Persing and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{persingq,vince}@hlt.utdallas.edu

Abstract

A poll consists of a question and a set of predefined answers from which voters can select. We present the new problem of vote prediction on comments, which involves determining which of these answers a voter selected given a comment she wrote after voting. To address this task, we exploit not only the information extracted from the comments but also extra-textual information such as user demographic information and inter-comment constraints. In an evaluation involving nearly one million comments collected from the popular SodaHead social polling website, we show that a vote prediction system that exploits only textual information can be improved significantly when extended with extra-textual information.

1 Introduction

We introduce in this paper a new opinion mining task, *vote prediction* on comments in social polls. Recall that a poll consists of a question accompanied by a set of predefined answers. A user who votes on the question will choose one of these answers and will be prompted to enter a comment giving an explanation of why she chose the answer. Given a poll and a user comment written in response to it, the task of *vote prediction* seeks to determine which predefined answer was chosen by the author of the comment.

A solution to the vote prediction problem would contribute significantly to our understanding of the underlying attitudes of individual social polling website users. This understanding could be exploited for tasks such as improving user experience or directed advertising; if we can predict how a user will vote on a question, we can make more accurate guesses about what kind of content/ads related to the question the user would like to see. Unfortunately, a major difficulty of vote prediction arises from the casual nature of discussion in social media. A comment often contains insufficient information for inferring the user's vote, or in some cases may even be entirely absent.

In light of this difficulty, we exploit two additional types of information in the prediction process. First, we employ demographic features derived from user profiles. Demographic features may be broadly useful for other opinion mining tasks such as stance classification (Somasundaran and Wiebe, 2010), as many social media websites like CreateDebate[1] allow users to create profiles with similar demographic information. Previous work has attempted to predict such latent features (e.g., Rao and Yarowsky (2010), Burger et al. (2011)) rather than employing them for opinion mining tasks.

Second, we exploit inter-comment constraints to help us perform joint inference over votes on different questions. Note that previous work on debate stance recognition has also employed constraints to improve the inference process. Specifically, in stance prediction, it is typical to employ so-called author constraints (e.g., Thomas et al. (2006), Bansal et al. (2008), Walker et al. (2012a), Hasan and Ng (2013)), which specify that two documents written by the same author for the same topic should have the same stance. However, in vote prediction, author constraints are not useful because a user is not permitted to cast more than one vote per question, unlike in stance prediction, where users may engage in a debate and therefore post more than once per debate topic. Consequently, we propose two new types of constraints for exploiting inter topic user voting patterns. One constraint involves pairs of authors and the other involves pairs of questions. These constraints are also potentially useful for other opin-

[1] http://www.createdebate.com/

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1127–1138,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ion mining tasks involving social media, as social media sites typically allow users to comment on multiple topics. Note that enforcing constraints involving two questions is by no means trivial, as the possible class values associated with the two comments may not necessarily be the same.

Another contribution of our work lies in our adaptation of the label propagation algorithm (Zhu and Ghahramani, 2002) to enforce constraints for vote prediction. Recall that existing stance classification approaches enforce constraints using minimum cut (Thomas et al., 2006), integer linear programming (Lu et al., 2012), and loopy belief propagation (Burfoot et al., 2011). Our decision to employ label propagation stems in part from the inability of loopy belief propagation and integer linear programming to efficiently process the nearly one million comments we have, and in part from the inability of the traditional two-way minimum cut algorithm to handle multiclass classification. It is worth noting, however, that other variations of the label propagation algorithm have been proposed for unrelated NLP tasks such as automatically harvesting temporal facts from the web (e.g., Wang et al. (2011) and Wang et al. (2012)).

While we are the first to address the vote prediction task, other researchers have previously used social media to predict the outcomes of various events, primarily by analyzing Twitter data. For example, Tumasjan et al. (2010) and Gayo-Avello et al. (2011) performed the related task of predicting the outcomes of elections. Rather than predicting election outcomes, O'Connor et al. (2010) focused on finding correlations between measures derived from tweets and the outcomes of political events like elections and polls. Finally, Asur and Huberman (2010) predicted movies' box office success. These tasks contrast with our task of vote prediction in that they are concerned with aggregate measures such as the fraction of the vote each candidate or party will win in an election or how much money a movie will make at the box office, whereas vote prediction is concerned with predicting how individual people will vote on a much wider variety of news/political topics.

2 Corpus

SodaHead[2] is a social polling website where users vote on and ask questions about a wide variety of topics ranging from the serious (e.g., "Should the U.S. raise the minimum wage?") to the silly (e.g "What is your favorite kind of pie?"). Whenever a user votes on one of these questions, choosing one of a set of predefined answers, she is prompted to enter a comment giving an explanation of why she chose the answer she did. Our corpus[3] consists of all the comments[4] users posted under all featured questions in the News & Politics category of the SodaHead website between March 12, 2008 and August 21, 2013.

This dataset consists of a total of 997,379 comments over 4,803 different questions, so an average of 208 comments are written in response to each question. The length of an average comment is 49 words. As Table 1 illustrates, these questions may have more than two possible answers, with an average question having 2.4 possible answers.

Each SodaHead user has her own profile that contains demographic information about her. As we can see from Table 2, many users choose to provide only some information about themselves, leaving many of the demographic fields blank. 108,462 users posted at least one comment in our corpus, with an average user commenting on 9.2 of our questions.

3 Baseline Systems

To perform our experiments, we first split our comments into three sets, a test set for evaluating performance, a training set for training classifiers, and a development set for tuning parameters. In order to ensure that the comparisons of our experiments are valid, we construct our test set using the same 20% of comments in the dataset regardless of experiment. Since our goal is to plot a learning curve illustrating how our various vote prediction systems perform given different amounts of training and development data, we vary the size of our training and development sets across experiments so that in the smallest experiment, together they comprise 25% of the remaining (non-test) comments, and in the largest experiment, they com-

[2] http://www.sodahead.com

[3] http://www.hlt.utdallas.edu/%7epersingq/SocialPolls/ is the distribution site for our corpus. We preserve user anonymity by replacing the original id of each user with a random number in our corpus.

[4] A "comment" is the text a user posted when submitting her vote on a question. It does not include posts not associated with a vote (such as responses to other posts) or votes where the user chose not to enter a comment. Thus, there is a one-to-one relationship between comments in votes in our dataset. The vote associated with a comment is always known.

Question	Vote	Comment
Who Won Round Two of the Presidential Debate?	Barack Obama	Binders full of women. That is all.
	Mitt Romney	Obama is inept and a liar. We can't survive 4 more years of his crazy crap.
What's the Best Way to Read a Magazine?	in print	Upside down like Luna Lovegood.
	online	Print costs money. It also doesn't have a Search function.
	on a tablet device	since sooooo many people have tablet devices why read it as print or online?
	on a smartphone	Clicked in print!!! Aargh

Table 1: Sample questions and comments. All of the pre-defined answers for these questions are represented by one comment.

User ID	3479864	3189372
Age	25-34	
Smoker		No
Drinker		No
Income		
Sexual Orientation		Straight
Relationship Status		Single
Political Views	Conservative	Moderate
Ethnicity		
Looking For		
Career Industry		
Children		Undecided
Education		High School
Gender	Female	Male
Religious Views	Other	Christian
Employment Status		
Weight Type		

Table 2: Sample user profiles.

prise 100% of the remaining comments. For each experiment, we maintain a ratio of three training comments to one development comment.

Recall that each comment in our dataset is written in response to a particular question. For each test comment, our goal is to predict the user's answer to the question given the text of her comment. One of the major inherent difficulties of our task is that it consists not of one, but of 4,803 separate multiclass classification problems (one for each question). As a result, our approach to the problem necessarily has to be somewhat generic, as it would be too time-consuming to develop an appropriate feature set for each question.

3.1 Baseline 1

Our first baseline's (B_1) approach employs 4,803 multiclass classifiers (one for each question). Each classifier is trained on one question's training set, representing each comment using only a bias feature. Each of our classifiers is trained using MALLET's (McCallum, 2002) implementation of maximum entropy (ME) classification. This is equivalent to merely counting the number of training set comments that voted for each possible answer, selecting the most frequent answer, then applying this label to all the comments in the test set. This majority baseline serves primarily to tell us how well our more sophisticated baseline performs.

3.2 Baseline 2

Our second baseline (B_2) is constructed in exactly the same way as B_1 except that each classifier is trained using both a bias feature and a standard set of feature types described below.

3.2.1 Features

Since the questions in our dataset come from the News & Politics category of the SodaHead website, many of the questions' topics are political. For that reason, it makes sense to use features which have been shown to work well on other political classification problems. We therefore base our feature set on that used by Walker et al. (2012b) for political debate classification. Our features are described below.

N-grams. Unigrams have been shown to perform well in ideological debates (Somasundaran and Wiebe, 2010), so we therefore present our classifiers with lemmatized unigram, bigram, and trigram features. We normalize the n-gram feature vector to unit length to avoid giving undue influence to longer comments.

Cue Words. Based on other work (Fox Tree and Schrock, 1999; Fox Tree and Schrock, 2002; Groen et al., 2010; Walker et al., 2012b), we also present our classifiers with features representing the first lemmatized unigram, bigram, and trigram appearing in each comment. These may be useful in our task when, for example, a user's comment begins with or entirely consists of a restatement of the answer she chose. So if the possible answers for a given question are "Yes" and "No", a user might write in her comment "Yes. Because ...", and this would make the "CueWord:Yes" feature useful for classifying this comment.

Emotion Frequency. For each word in a comment, we used the NRC Emotion Word Lexicon

(Mohammad and Yang, 2011) to discover if the word conveys any emotion. Then, for each emotion or sentiment covered by the lexicon (anger, anticipation, disgust, fear, joy, sadness, surprise, trust, positive, or negative) e_i, we construct a feature e_i:$\frac{C(e_i)}{total}$ describing how much of the comment consists of words conveying emotion e_i, where $C(e_i)$ is the count of words in the comment bearing emotion e_i and $total$ is the number of words in the comment. To understand why this feature may be useful, consider the question "Does Sarah Palin deserve VP?" We suspect that users who post comments laden with words associated with positive emotions like joy are more likely to vote "Yes" because the positive emotions imply they are happy about a Sarah Palin vice presidency. Similarly, users who post comments laden with negative emotions like anger might be more likely to vote "No".

Dependencies. We use the Stanford Parser (de Marneffe et al., 2006) to extract a set of dependencies from each comment. For an example of how dependencies might help in our task, consider the second comment in Table 1. From this comment, we can extract the dependency triple dependency:(nsubj,inept,obama), which indicates that the user who wrote it does not like Obama and is therefore more likely to have voted for Romney in the question. Dependency feature vectors are normalized to unit length.

Emotion Dependencies. To form an emotion dependency feature, we take a regular dependency feature and replace each of its words where possible with the emotion it evokes as determined by the NRC Emotion Word Lexicon. Thus from the dependency:(nsubj,inept,obama) example above, we would generate three features: emotiondependency:(nsubj,anger,obama), emotiondependency:(nsubj,disgust,obama), and emotiondependency:(nsubj,negative,obama). These features help generalize dependencies, and this is useful because predictive features like emotiondependency:(nsubj,negative,obama) appear frequently in the comments for this question, but dependency:(nsubj,inept,obama) does not. Emotion Dependency feature vectors are normalized to unit length.

Post Information. Features under this category just calculate some basic statistics about a comment. These features may be useful because, for example, the question "Most Scandalous Politicians of 2008− Who deserves the title?" has six possible answers, each except the last naming a particular well-known politician. The last choice is "The most scandalous politician of 2008 is ..." and the user is expected to name a politician in her comment. It would make sense for users choosing this option to have written longer responses since they have to name and possibly explain their choice to users who might not necessarily know who their chosen politician is.

3.2.2 Feature Selection

Because some of the feature types (n-grams, cue words, dependencies, and emotion dependencies) described in the previous subsection are expected to generate a large number of non-predictive features, we trim some of the most irrelevant features out of the feature set to avoid memory problems. Therefore, following Yang and Pedersen (1997), for each question we calculate the information gain of each feature of these types on the training set. We then remove those features having the lowest information gain as well as those features occurring less than ten times in the dataset. Early experiments showed that 1,000 was a reasonable number of features to keep, so for all experiments we keep only the top 1,000 features of these types. Note that we do not apply feature selection to emotion frequency or post information features, as each of these sets consists of a small number of real-valued features.

4 Demographic Features

As mentioned in the introduction, a major difficulty inherent to our problem is that in many cases a comment contains insufficient information for inferring the underlying vote. Aside from being short, the comments shown in Table 1 are typical of comments found in the dataset. Some comments are like the first and third in the table, requiring some obscure bit of world knowledge to understand what the writer is saying. Others like the fourth only explain why the user did not choose a particular answer, which is always potentially useful, but sufficient only if the comment excludes every other possible choice.

Because it is difficult to tell how a user voted given her comment, we exploit the demographic information users provide in their profiles as an additional source of information. Since many of the questions in our dataset deal with politics, we anticipate that information about things such as whether a comment was written by a conservative or progressive user would be useful for predicting the answers of many comments. For each comment, we encode demographic information as features in the following way. For each field in the user's profile shown in Table 2 (aside from user ID), we construct a feature of the form F_i:V_i if the user filled in field F_i with value V_i. Thus, any comment made by user 3479864 would include the features Age:25−34, PoliticalViews:Conservative, Gender:Female, and Religion:Other.

Here is an example of a comment whose predicted vote gets corrected by adding demographic features to our system. For the question, "LPGA Decides to Allow Transgender Competitors: Good or Bad Move for Golf?", user 2252750 writes, "LPGA ...can let monkeys play if they wish....nobody gives a rip... bark". Of the three possible answers for this question, "Good move", "Bad move", and "Undecided", our baseline system without demographics believes that user 2252750 probably voted for the third, as "nobody gives a rip" makes him sound apathetic toward the issue. However, our demographic system notices that his profile contains "Religion:Christian", and users with this demographic attribute choose "Bad Move" 64% of the time. Thus, demographic features allowed our system to correctly predict his vote for "Bad Move".

Since demographics are also expected to generate a large number of non-predictive features, we apply feature selection to them as described in Section 3.2.2.

5 Enforcing Constraints

We mentioned earlier that an average SodaHead question contains 208 comments. This implies that there are only about 31−125 comments[5] in the average training set for one of our ME classifiers. It would be difficult to train a good classifier from a training set this small even if we had feature sets tailored to work well on each of the 4,803 questions. While we have already attempted to exploit user information (in the form of demographic features) to help improve our system's performance, this approach still treats the task as 4,803 separate classification problems. It does not allow for the possibility that classification on one question may be improved by exploiting information gleaned from votes on other questions.

One way we might exploit such information is by first noticing that, for any pair of questions, there may be multiple users who commented on both. This overlap between questions allows us to calculate how predictive a user's vote on one question is of how she will vote on the other. For example, on the question "Who Would You Rather Have Dinner With?", we found that users who voted for "Mitt Romney" were much more likely to choose "No, I'm still voting for him" on the question "Does Mitt Romney's 'Entitled' Remarks Change Your Opinion of Him?". Similarly, users who voted to have dinner with "Barack Obama" were much more likely to vote "Yes, I'm not voting for him anymore" on the "entitled" question. A system that somehow takes into account this information might correctly classify a difficult comment on the "entitled" question if it notices that the comment was written by a user who commented on both questions and it knows how the user voted on the "dinner" question. We call the kind of constraint described here a **QuestionPair** constraint.

We might also exploit information from other questions by noticing that there are users who share similar attitudes on a wide variety of topics in our dataset. We can gauge how often a pair of users agree with each other by comparing their votes on every question on which they have both voted where their comments appear in the training set. So for example, if we see that two users have agreed on questions about George H.W. Bush, Bill Clinton, and George W. Bush, we can guess that they will also agree on a question about Barack Obama. Similarly, if they disagreed on all those questions, they are likely to disagree on the last question. A system that takes into account this kind of information could correctly classify an otherwise difficult comment if it knows how another user voted on this question and also knows how often the two users agree on other questions. We call the kind of constraint described here a **VoterPair** constraint.

In order to enforce both kinds of constraints, we introduce a variation of the label propagation algorithm (Zhu and Ghahramani, 2002). In our version of the label propagation algorithm, each comment in our dataset is represented by a node in a graph. Each node is associated with a probability distribution indicating the likelihood that the

[5] At the low and high end of the learning curve respectively.

comment belongs to each of its question's possible answers. Thus, when we initialize the graph, each training set node's probability distribution is set to reflect its comment's actual label (with a probability of 1 for the comment's actual label and 0 for each other answer), and each development or test set node's probability distribution is set to the value predicted by another classifier such as B_2 or $B_2 + Dem$ since the algorithm is not permitted to see the comment's actual label. Lines $7-12$ in Figure 1 describe the graph's initialization.

Now that we have set up the graph's nodes, we need to explain how our graph's edges work. As we discussed earlier in this section, the edges in our graph will represent two kinds of soft constraints. Each edge allows one of a node's neighbors to cast a vote (in the form of a probability distribution over possible answers) for what it thinks the node's answer should be. Let us call the comment node whose label we are trying to predict the **target** node and the comment node which casts the vote the **source** node.

Our graph contains a QuestionPair edge between any source and target comments written by the same user. Since a user cannot comment more than once on any question, the source and target comments will occur in two different questions. In order to determine how the source node votes over a QuestionPair edge, we need to calculate some probabilities. In particular, we need to determine the probability that a user will vote for possible answer k in the target question Q_I given that she voted for answer l in the source question Q_J:

$$P(Q_{I_k}|Q_{J_l}) = \frac{C(Q_{I_k},Q_{J_l})+\gamma}{\sum\limits_{m\in A(Q_I)}(C(Q_{I_m},Q_{J_l})+\gamma)}$$

where $C(Q_{I_n}, Q_{J_l})$ is the number of users who voted for answer n in Q_I and answer l in Q_J, and $A(Q_I)$ is the set of possible answers on Q_I. We set γ, the smoothing factor, to 10 since this value worked well in earlier experiments. The source node S casts its vote on target node T for the probability distribution given by:

$$QP_{S,T}(Q_{I_k}) = \sum_{m\in A(Q_J)} P_S(Q_{J_m})P(Q_{I_k}|Q_{J_m})$$

where $P_S(Q_{J_m})$ is the probability currently associated with answer m in S's question (Q_J).

The graph contains a VoterPair edge between any source and target nodes on the same question if the users who posted these comments have both voted on at least one other question together and their comments on the other question(s) occurred in the training set. To determine how the source node votes over a VoterPair edge, we need to calculate the probability that the source and target users will agree on a generic issue:

$$P_{agr}(U_S,U_T) = \frac{C_{agr}(U_S,U_T)+1}{C_{agr}(U_S,U_T)+C_{dis}(U_S,U_T)+2}$$

where $C_{agr}(U_S,U_T)$ is the number of questions on which users U_S and U_T voted for the same answer and both their comments occurred in the training set, $C_{dis}(U_S,U_T)$ is the number of questions on which U_S and U_T voted for different answers where both their comments occurred in the training set, and the $+1$ and $+2$ are used for smoothing. The probability distribution that the source node S votes for on target node T is then given by:

$$VP_{S,T}(Q_{I_k}) = P_S(Q_{I_k})P_{agr}(U_S,U_T) + \sum_{\substack{m\in A(Q_I),\\ m\neq k}} (P_S(Q_{I_m}))\frac{1-P_{agr}(U_S,U_T)}{|A(Q_I)|-1}$$

where $P_S(Q_{I_n})$ is the probability currently associated with answer n in the source node's question (Q_I), and $|A(Q_I)|$ is the number of possible answers on Q_I. We divide the second term, which deals with disagreement, by $|A(Q_I)|-1$ because, even if we know that the target and source users disagreed on the answer to a particular question and that the source user did not vote for answer k, there is only a $\frac{1}{|A(Q_I)|-1}$ chance that the target user voted for answer k since there are $|A(Q_I)|-1$ non-k answers to choose from.

Now that we have described how edges are added to the graph and how source comment nodes vote over the edges, we are ready to begin iterating over the label propagation algorithm (line 13 in Figure 1). For each iteration of the algorithm, we update each development or test set node's answer probability distribution by assigning it a weighted sum of (1) the initial probability distribution assigned to the node, (2) the sum of the QuestionPair edges' votes, and (3) the sum of the VoterPair edges' votes (line 16 in Figure 1). Upon completion of the algorithm, if our soft constraints work as expected, the new labeling of comment nodes should be more accurate than their initial labeling.

We tune the parameters W_I, W_V, W_Q, and $iterations$ jointly by an exhaustive search of the parameter space to maximize classification accuracy on the development set. Each of the weight parameters is allowed to take one of the values 0, 1, or 2, and the iteration parameter is allowed take one of the values 0, 1, 2, 3, 4, 5.

```
1: $LabelPropagation(Tr, D, Te, iterations, W_i, W_V, W_Q, I)$
2: **Inputs:**
3:     $Tr, D, Te$: Comments in Training, Development, and Test set
4:     $iterations$: The number of iterations to perform
5:     $W_i, W_V, W_Q$: Weights assigned to initial, VoterPair, and QuestionPair constraints
6:     $I$: Initial answer probability distribution for all comments. Should reflect actual labels for training set comments and
   classifier predictions for development and test set comments
7: **for all** $C \in Tr \cup D \cup Te$ **do**
8:     Create node representing C
9:     $C_p \leftarrow I_C$
10:    // $C_p$: node $C$'s current probability distribution over possible answers
11:    // $I_C$: initial answer probability distribution for comment C
12: **end for**
13: **for** $j = 1$ to $iterations$ **do**
14:    **for all** node $C \in D \cup Te$ **do**
15:        Add all edges targeting node $C$
16:        $C_p \leftarrow Norm(W_I I_C + W_V \sum_k VP_{k,C} + W_Q \sum_k QP_{k,C})$
17:        // $VP_{k,C}$, $QP_{k,C}$: $k$th VoterPair, and $k$th QuestionPair votes for node C
18:        Remove all edges targeting node $C$
19:    **end for**
20: **end for**
```

Figure 1: Our label propagation algorithm.

One may be surprised to notice how we add edges to the graph in the algorithm only to delete them three lines later (lines 15 and 18 in Figure 1). Though edges can be added at any point in the algorithm, one benefit of using the label propagation algorithm is that it is simple enough that it is not necessary store all the edges in memory at once. The only time we need to store an edge is when its target is being voted on. This means that the label propagation algorithm can handle large datasets like ours with huge numbers of nodes and edges without being prohibitively space-expensive.

6 Evaluation

6.1 Experimental Setup

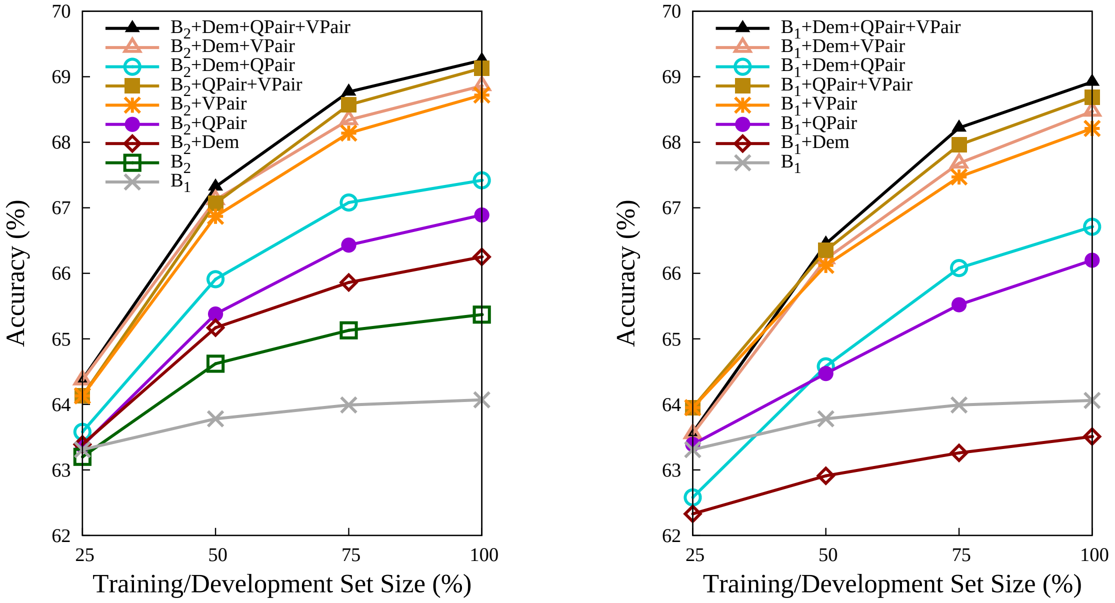
We mentioned in Section 3 that we split our dataset of 997,379 comments into a test set comprising about 20% of the dataset's comments and a training and development set comprising some fraction of the remaining 80% of the comments. We actually split the data up like this five different times so that each comment appears in an experiment's test set exactly once. In this way, through the use of five fold cross-validation, we can report our results on the entire dataset.

6.2 Results and Discussion

Figure 2a shows the accuracy of the predictions made by various systems. First, let us compare our first and second baselines. Recall that the first baseline (B_1) predicts that all test comments will have the same label as the majority of training comments, and the second baseline's (B_2) predictions are the output of ME classifiers trained with a generic feature set. As we can see from the graph, at very small training set sizes, the standard set of features supplied to B_2 does little more than confuse the ME learner, as it performs slightly but not significantly worse[6] than the first baseline when the training/development set comprises only 25% of the available data. This is understandable, as 25% of an average question's available data is only 42 comments, an extremely small number of examples to learn from for most NLP tasks. Clearly a better approach than the one provided by the second baseline is needed. Though the average training set sizes at the 50%, 75%, and 100% levels are still relatively small, B_2 significantly outperforms B_1 at all these levels.

The small improvement sizes yielded by B_2 may be attributable to some of the inherent difficulties of the problem, particularly that (1) it is composed of so many (4,803) separate subproblems that it is impractical for us to tailor a unique feature set for each one, (2) the average question is associated with a very small number of comments (about 208), making it difficult to train a reasonably good classifier for any question, and (3) many of the comments contain insufficient information for inferring the underlying votes. Perhaps some of our proposed extensions to B_2 can help address

[6] All significance tests are paired t-tests, with $p < 0.05$. Because we calculate a large number of significance results, the p values we report are obtained using Holm-Bonferroni multiple testing correction (Holm, 1979).



(a) Vote prediction.

(b) Arbitrary User Vote Prediction.

Figure 2: Five-fold cross-validation vote prediction learning curves.

some of these problems.

The first improvement we proposed involved exploiting demographic features provided by users to help with our prediction tasks. When we combine Dem and B_2's feature sets, the resulting system ($B_2 + Dem$) performs better than any of the systems discussed thus far at all four training/development set size levels, yielding significant improvements over B_2 at all four levels. This demonstrates that our demographic features are a useful complement to a standard approach like the one used by B_2.

The second improvement we proposed involved using a variation of the label propagation algorithm to enforce QuestionPair constraints. QuestionPair constraints, recall, allowed us to exploit the observed voting patterns of users who voted in the training set on any particular pair of questions. These constraints were expected to improve our predictions for any user who voted on both questions when at least one of their votes appeared in the test set. System $B_2 + QPair$ corresponds to following the algorithm in Figure 1, using system B_2's ME classifiers to initialize a label propagation graph, and then setting the VoterPair edge weight (W_V) to 0, thus allowing only QuestionPair constraints. When we compare this system to B_2, we see that the performance boost QuestionPair constraints give us over the baseline is consistently greater than the boost given by adding demographic features to it ($B_2 + Dem$) across all training/development set sizes. The improvement over B_2 is even significant at the 75% and 100% training/development set sizes.

The last improvement we proposed involved adding VoterPair constraints to the label propagation graph. Recall that VoterPair constraints allowed us to exploit how frequently we observed two users agreeing with each other to predict whether they will agree on any question they both voted on. System $B_2 + VPair$ corresponds to following the label propagation algorithm using B_2's ME classifiers to initialize the graph, then setting the QuestionPair edge weight (W_Q) to 0, thus allowing only VoterPair constraints. The addition of VoterPair constraints yields the largest significant improvements over B_2 at all four levels, indicating that, in the absence of our other proposed improvements, VoterPair edge constraints are the most important addition we can make to our baseline.

While we have now shown that each of our proposed extensions yields significant improvements over B_2, this does not necessarily mean that each one is useful in the presence of the others. For example, it might be the case that QuestionPair constraints and Demographic features correct the same kinds of classification errors, and therefore it may be sufficient to use either one or the other to obtain good results, but using both is unnecessary. To test how useful they are in each other's presence, we perform the following experiment. First, we run the algorithm using all three improvements ($B_2 + Dem + QPair + VPair$ in Figure 2a). We then run the same experiment three more times, each time removing one of the three extensions. By measuring how much performance decreases when we remove each of the three improvements, we can determine whether each improvement provides unique useful information, or whether the information it provides is already being provided by one of the other improvements.

To see what happens when we remove demographic features from the full system, we need to compare $B_2 + Dem + QPair + VPair$ and $B_2 + QPair + VPair$ in Figure 2a. While the decrease in performance after removing demographic features was modest, the difference is nevertheless significant at all four training/development set sizes, suggesting that demographic features do provide unique information to the system.

By comparing line $B_2 + Dem + QPair + VPair$ to line $B_2 + Dem + VPair$, we can determine the impact of QuestionPair constraints. Removing QuestionPair constraints also had a modest impact on the full system's performance, decreasing accuracy at all four training/development set sizes, significantly so at the 50%, 75%, and 100% levels. Interestingly, the impact of QuestionPair constraints appears to grow with the training set, while the demographic features appear to have a greater impact when the training set is small. We can see this by noting that the two lines cross at around 55%. This suggests that QuestionPair constraints are especially useful in problems where it is cheap to obtain a lot of training data, but in problems where the data has to be manually annotated, demographic features are more useful.

Finally, we can compare line $B_2 + Dem + QPair + VPair$ to line $B_2 + Dem + QPair$ to see what happens when we remove VoterPair constraints from our system. This comparison illustrates that VoterPair constraints are by far the most important improvement we removed from the full system, as removing them yielded large significant decreases at all four levels.

Though thus far we have only used it to analyze the the contributions of different individual improvements, the full system $B_2 + Dem + QPair + VPair$ is interesting in itself. Of all the systems we have constructed, it performs the best, yielding improvements of up to 5.18% and 3.88% when compared to B_1 and B_2 respectively. Its improvements over both baselines are statistically significant at all four training/development set sizes.

6.3 Arbitrary User Vote Prediction

One interesting question that we have not yet addressed is, is it possible to predict how a user would vote on a question she has not yet seen? This problem is interesting because an average question receives votes from only 0.2% of the users in our dataset, and thus a system for predicting an arbitrary user's vote would be able to predict the votes of the other 99.8% of users. A solution to this prediction problem would have practical applications in areas such as directed advertising (e.g., if we could predict how a user would vote on the magazine question in Table 1, we would have a better idea of what kinds of reading devices/services would interest her).

We can mimic this problem with our dataset by treating the comment text associated with test votes as unseen since we cannot expect an arbitrary user to have commented on any particular question we are interested in[7]. It does, however, make sense for us to expect our arbitrary user to have provided some personal demographic information, and thus a system for making these types of predictions could reasonably make use of demographic features. Similarly, in this situation we would expect to have knowledge of all users' training set voting histories. Thus, it would also be reasonable for our system to exploit the QuestionPair and VoterPair constraints described in Section 5. Thus, to test how well our system performs on this task, we repeat all experiments from the previous section while replacing B_2 (which uses a ME classifier trained on comment-based features) with B_1 (the most frequent baseline, which uses a ME classifier trained using only a bias feature). The results of these experiments are shown in Figure 2b.

If we compare the results from B_1 to $B_1 + Dem$ (which compliments B_1's bias feature with the demographic feature set), we notice that $B_1 + Dem$ is significantly worse than B_1 at all training set sizes. This confirms our suspicion from the pre-

[7] Although we are trying to mimic the situation in which we predict how an arbitrary user would vote on an arbitrary question, we caution that the vote data we train and evaluate on was not obtained from a set of arbitrary SodaHead users. It consists only of votes from users who chose which questions they wanted to answer. For this reason, the data we train and evaluate on for any question might not be a representative sample of SodaHead users as a whole.

vious section that demographic features by themselves serve only to confuse the learner, though we will see in a moment that they are a helpful supplement to more sophisticated systems.

We can evaluate QuestionPair constraints in this setting by comparing the results from B_1 to $B_1 + QPair$. $B_1 + QPair$ consistently outperforms B_1 at all four training set sizes, significantly so at the 75% and 100% levels, and thus QuestionPair constraints are also a useful addition to our system.

VoterPair constraints can be evaluated in this setting by comparing B_1 to $B_1 + VPair$. $B_1 + VPair$ significantly outperforms B_1 at all four training set sizes, and from the graph it appears to be our most beneficial improvement.

To evaluate whether demographic features are useful in the presence of the other improvements, we compare the full system, $B_1 + Dem + QPair + VPair$, to its corresponding version without demographic features, $B_1 + QPair + VPair$. Though $B_1 + QPair + VPair$ significantly outperforms the full system at the 25% training set size, the full system significantly outperforms $B_1 + QPair + VPair$ at the 75% and 100% levels, indicating that in this setting, demographic features are useful in the presence of a large training set.

We can evaluate the utility of QuestionPair constraints in this setting by comparing the full system to $B_1 + Dem + VPair$. When we remove QuestionPair constraints, accuracy is consistently lowered at all four training set sizes, significantly so at 50%, 75%, and 100%. This tells us that QuestionPair constraints are useful in this setting.

We can evaluate how useful VoterPair constraints are by checking how much $B_1 + Dem + VPair + QPair$'s performance drops when we remove VoterPair constraints from it, yielding $B_1 + Dem + QPair$. Performance drops considerably and significantly at all four training set sizes after removing VoterPair constraints, suggesting that in this setting, VoterPair constraints are still the most important of our proposed improvements.

Finally, while we have already established that all our proposed improvements can improve performance under both settings (comments visible and comments invisible), it may be worthwhile to compare the two sets of experiments to determine whether the comment features used in systems with B_2 are useful.

A casual inspection of the two figures shows that, broadly, each system that uses comment-based features in Figure 2a tends to slightly outperform the most comparable system in Figure 2b. At the low end of the curves, the two systems often differ by about 1.0% in absolute accuracy, though at the high end, the difference tends to be much smaller, with the full system with comment features outperforming the full system without comment features by only 0.3%. Since in this setting it is reasonable to assume a large training set, this last result is the one we are most interested in, and it suggests that our full system's performance does not suffer much due to the absence of comment features.

One final observation we can make is that, when comments are not visible, demographic features appear to actively harm the performance of systems trained on a small amount of data, though at larger training set sizes they are mostly helpful. We can tell this by comparing systems with demographic features to systems without them in Figure 2b (e.g., by comparing $B_1 + Dem + QPair$ to $B_1 + QPair$ or $B_1 + Dem + VPair$ to $B_1 + VPair$) at the 25% training set size. This is not the case in the setting where comments are visible, as we see that demographic features always appear helpful in Figure 2a. This reinforces the notion that demographic features provide useful information in general, but that they are by themselves too sparsely available to do more than confuse the learner. They need to be supplemented by other information sources in order for the learner to draw correct conclusions.

7 Conclusion

We examined the task of vote prediction on comments from the SodaHead website. To address this task, we exploited not only information extracted from the comments but also extra-textual information, including demographic information and two types of inter-comment constraints, QuestionPair constraints and VoterPair constraints. Our experiments involving 997,379 comments showed that each of these extensions significantly improved a baseline that exploited only textual information, with VoterPair constraints being the most effective and demographic information being the least effective. When used in combination, they obtained up to a 3.88% improvement in absolute accuracy over the baseline. To stimulate research on this task, we make our dataset publicly available.

Acknowledgments

We thank the three anonymous reviewers for their detailed and insightful comments on an earlier draft of this paper. This work was supported in part by NSF Grants IIS-1147644 and IIS-1219142. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Sitaram Asur and Bernardo A. Huberman. 2010. Predicting the future with social media. In *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 492–499.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *COLING 2008: Companion Volume: Posters*, pages 15–18.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1506–1515.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Gender discrimination on twitter. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1301–1309.

Marie-Catherine de Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 449–454.

Jean E. Fox Tree and Josef C. Schrock. 1999. Discourse markers in spontaneous speech: Oh what a difference an oh makes. *Journal of Memory and Language*, 40:280–295.

Jean E. Fox Tree and Josef C. Schrock. 2002. Basic meanings of you know and i mean. *Journal of Pragmatics*, 34:427–447.

Daniel Gayo-Avello, Panagiotis Takis Metaxas, and Eni Mustafaraj. 2011. Limits of electoral predictions using twitter. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 490–493.

Martin Groen, Jan Noyes, and Frans Verstraten. 2010. The effect of substituting discourse markers on their role in dialogue. *Discourse Processes: A Multidisciplinary Journal*, 47:388–420.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65–70.

Yue Lu, Hongning Wang, ChengXiang Zhai, and Dan Roth. 2012. Unsupervised discovery of opposing opinion networks from forum discussions. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1642–1646.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Saif Mohammad and Tony Yang. 2011. Tracking sentiment in mail: How genders differ on emotional axes. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, pages 70–79.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 122–129.

Delip Rao and David Yarowsky. 2010. Detecting latent user properties in social media. In *Proceedings of the NIPS workshop on Machine Learning for Social Networks*.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

Andranik Tumasjan, Timm Sprenger, Philipp Sandner, and Isabell Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media*, pages 178–185.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012a. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596.

Marilyn A. Walker, Pranav Anand, Rob Abbott, Jean E. Fox Tree, Craig Martell, and Joseph King. 2012b. That is your evidence?: Classifying stance in online political debate. *Decision Support Systems*, 53(4):719–729.

Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. 2011. Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 837–846.

Yafang Wang, Maximilian Dylla, Marc Spaniol, and Gerhard Weikum. 2012. Coupling label propagation and constraints for temporal fact extraction. In *Proceedings of the ACL 2012 Conference Short Papers*, pages 233–237.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning*, pages 412–420.

Xiaojin Zhu and Zoubin Ghahramani. 2002. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, CMU CALD.