Crystal: Analyzing Predictive Opinions on the Web

Soo-Min Kim and Eduard Hovy
USC Information Sciences Institute
4676 Admiralty Way, Marina del Rey, CA 90292
{skim,hovy}@ISI.EDU

Abstract

In this paper, we present an election prediction system (*Crystal*) based on web users' opinions posted on an election prediction website. Given a prediction message, *Crystal* first identifies which party the message predicts to win and then aggregates prediction analysis results of a large amount of opinions to project the election results. We collect past election prediction messages from the Web and automatically build a gold standard. We focus on capturing lexical patterns that people frequently use when they express their predictive opinions about a coming election. To predict election results, we apply SVM-based supervised learning. To improve performance, we propose a novel technique which generalizes n-gram feature patterns. Experimental results show that *Crystal* significantly outperforms several baselines as well as a non-generalized n-gram approach. *Crystal* predicts future elections with 81.68% accuracy.

1 Introduction

As a growing number of people use the Web as a medium for expressing their opinions, the Web is becoming a rich source of various opinions in the form of product reviews, travel advice, social issue discussions, consumer complaints, stock market predictions, real estate market predictions, etc.

At least two categories of opinions can be identified. One consists of opinions such as "I like/dislike it", and the other consists of opinions like "It is likely/unlikely to happen." We call the first category ***Judgment Opinions*** and the second (those discussing the future) ***Predictive Opinions***. Judgment opinions express positive or negative sentiment about a topic such as, for example, reviews about cameras, movies, books, or hotels, and discussions about topics like abortion and war. In contrast, predictive opinions express a person's opinion about the future of a topic or event such as the housing market, a popular sports match, and national election, based on his or her belief and knowledge.

Due to the different nature of these two categories of opinion, each has different valences. Judgment opinions have core valences of *positive* and *negative*. For example, "liking a product" and "supporting abortion" have the valence "positive" toward each topic (namely "a product" and "abortion"). Predictive opinions have the core valence of *likely* or *unlikely* predicated on the event. For example, a sentence "Housing prices will go down soon" carries the valence of "likely" for the event of "housing prices go down".

The two types of opinions can co-appear. The sentence "I like Democrats but I think they are not likely to win considering the war issue" contains both types of opinion: "positive" valence towards Democrats and "unlikely" valence towards the event of "Democrats wins". In order to accurately identify and analyze each type of opinion, different approaches are desirable.

Note that our work is different from predictive data mining which models a data mining system using statistical approaches in order to forecast the future or trace a pattern of interest (Rickel and Porter, 1997; Rodionov and Martin, 1996). Example domains of predictive data mining include earthquake prediction, air temperature prediction, foreign exchange prediction, and energy price predic-

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 1056–1064, Prague, June 2007.

tion. However, predictive data mining is only feasible when a large amount of structured numerical data (e.g., in a database) is available. Unlike this research area which analyzes numeric values, our study mines unstructured text using NLP techniques and it can potentially extend the reach of numeric techniques.

Despite the vast amount of predictive opinions and their potential applications such as identification and analysis of people's opinions about the real estate market or a specific country's economic future, studies on predictive opinions have been neglected in Computational Linguistics, where most previous work focuses on judgment opinions (see Section 2). In this paper, we concentrate on identifying predictive opinion with its valence.

Among many prediction domains on the Web, we focus on election prediction and introduce *Crystal*, a system to predict election results using the public's written viewpoints. To build our system, we collect opinions about past elections posted on an election prediction project website before the election day, and build a corpus[1]. We then use this corpus to train our system for analyzing predictive opinion messages and, using this, to predict the election outcome. Due to the availability of actual results of the past elections, we can not only evaluate how accurately *Crystal* analyzes prediction messages (by checking agreement with the gold standard), but also objectively measure the prediction accuracy of our system.

The main contributions of this work are as follows:

- an NLP technique for analyzing predictive opinions in the electoral domain;
- a method of automatically building a corpus of predictive opinions for a supervised learning approach; and
- a feature generalization technique that outperforms all the baselines on the task of identifying a predicted winning party given a predictive opinion.

The rest of this paper is structured as follows. Section 2 surveys previous work. Section 3 formally defines our task and describes our data set. Section 4 describes our system *Crystal* with proposed feature generalization algorithm. Section 5 reports empirical evidence that *Crystal* outperforms several baseline systems. Finally, Section 6 concludes with a description of the impact of this work.

[1] The resulting corpus is available at
http://www.isi.edu/ ~skim/Download/Data/predictive.htm

2 Related Work

This work is closely related to opinion analysis and text classification. Most research on opinion analysis in computational linguistics has focused on sentiment analysis, subjectivity detection, and review mining. Pang et al. (2002) and Turney (2002) classified sentiment polarity of reviews at the document level. Wiebe et al. (1999) classified sentence level subjectivity using syntactic classes such as adjectives, pronouns and modal verbs as features. Riloff and Wiebe (2003) extracted subjective expressions from sentences using a bootstrapping pattern learning process. Wiebe et. al (2004) and Riloff et. al (2005) adopted pattern learning with lexical feature generalization for subjective expression detection. Dave et. al (2003) and Jindal and Liu (2006) also learned patterns of opinion expression in product reviews. Yu and Hatzivassiloglou (2003) identified the polarity of opinion sentences using semantically oriented words. These techniques were applied and examined in different domains, such as customer reviews (Hu and Liu 2004; Popescu et al., 2005) and news articles (Kim and Hovy, 2004; Wilson et al., 2005).

In text classification, systems typically use bag-of-words models, mostly with supervised learning algorithms using Naive Bayes or Support Vector Machines (Joachims, 1998) to classify documents into several categories such as sports, art, politics, and religion. Liu et al. (2004) and Gliozzo et al. (2005) address the difficulty of obtaining training corpora for supervised learning and propose unsupervised learning approaches. Another recent related classification task focuses on academic and commercial efforts to detect email spam messages. For an SVM-based approach, see (Drucker et al., 1999). In our study, we explore the use of generalized lexical features for predictive opinion analysis and compare it with the bag-of-words approach.

3 Modeling Prediction

In this section, we define the task of analyzing predictive opinions in the electoral domain.

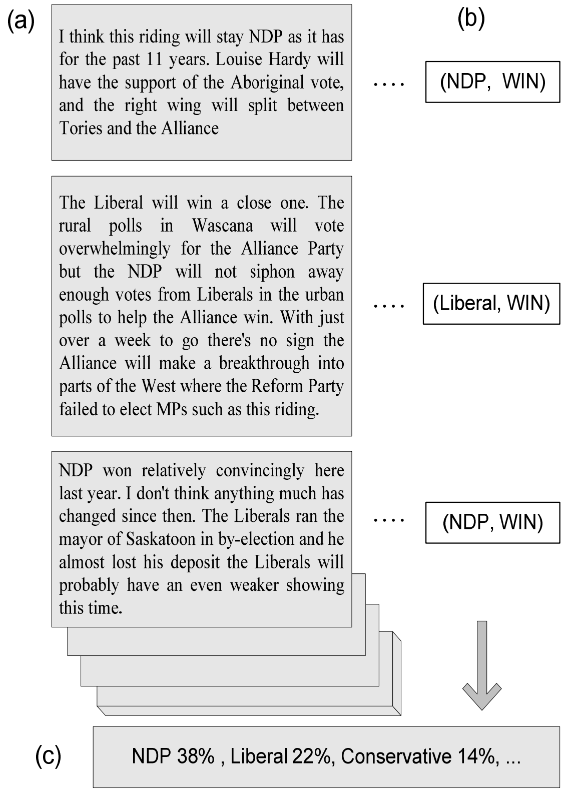

Figure 1. Our election prediction system. Public opinions are collected from message boards (a) and our system determines for each the election prediction 'Party' and 'Valence' (b). The output of the system is a prediction of the election outcome (c).

3.1 Task Definition

We model predictive opinions in an election as follows:

$$ElectionPredictionOpinion = (Party, Valence)$$

where *Party* is a political party running for an election (e.g., Democrats and Republicans) and *Valence* is the valence of a predictive opinion which can be either *"likely to win"* (***WIN***) or *"unlikely to win"* (***LOSE***). Values for *Party* vary depending on in which year (e.g., 1996 and 2006) and where an election takes place (e.g., United States, France, or Japan). The unit of a predictive opinion is an unstructured textual document such as an article in a personal blog or a message posted on a news group discussion board about the topic of "Which party do you think will win/lose in this election?".

Message text	Predicted winning party	Riding	Year
...	...	...	...
Message_1457	Party_3	Riding_206	2004
Message_1458	Party_2	Riding_206	2004
Message_1459	Party_2	Riding_189	2006
Message_1460	Party_1	Riding_189	2006
Message_1461	Party_2	Riding_189	2006
Message_1462	Party_1	Riding_46	2006
...	...	...	...

Table 1. A snapshot of the processed data

Riding name	Party	Candidate name
Blackstrap	NDP	Noreen Johns
	Liberal	J. Wayne Zimmer
	PC	Lynne Yelich

Table 2. An example of our Party-Candidate listing for a riding (PC: Progressive Conservative)

Figure 1 illustrates an overview of our election prediction system *Crystal* in action. Given each document posted on blogs or message boards (e.g., www.election prediction.org) as seen in Figure 1.a, a system can determine a *Party* that the author of a document thinks to win or lose (*Valence*), Figure 1.b. For the example document starting with the sentence "I think this riding will stay NDP as it has for the past 11 years." in Figure 1.a, our predictive opinion analysis system aims to recognize NDP as *Party* and *WIN* as *Valence*. After aggregating the predictive opinion analysis results of all documents, we project the election results in Figure 1.c. The following section describes how we obtain our data set and the subsequent sections describe *Crystal*.

3.2 Automatically Labeled Data

We collected messages posted on an election prediction project page, www.electionprediction. org. The website contains various election prediction projects (e.g., provincial election, federal election, and general election) of different countries (e.g., Canada and United Kingdom) from 1999 to 2006. For our data set, we downloaded Canadian federal election prediction data for 2004 and 2006. The Canadian federal electoral system is based on 308 ridings (electoral districts). The website contains 308 separate html files of messages corresponding to the 308 ridings for different years. In total, we collected 4858 and 4680 messages for the 2004 and 2006 federal elections respectively. On average, a message consists of 98.8 words.

To train and evaluate our system, we require a gold standard for each message (i.e., which party does an author of a message predict to win?). One option is to hire human annotators to build the gold standard. Instead, we used an online party logo image file that the author of each message already labeled for the message. Note that authors only select parties they think will win, which means our gold standard only contains a party with *WIN* valence of each message. However, we leverage this information to build a system which is able to determine a party even with *LOSE* valence. We describe this idea in detail in Section 4.

Finally, we pre-processed the data by converting the downloaded html source files into a structured format with the following fields: *message, party, riding*, and *year*, where *message* is a text, *party* is a winning party predicted in the text, *riding* is one of the 308 ridings, and *year* is either 2004 or 2006. Table 1 shows a snapshot of the processed data set that we used for our system training and evaluation. An additional piece of information consisting of a candidate's name for each party for each riding was also stored in our data set. With this information, the system can infer opinions about a party based on opinions about candidates who run for the party. Table 2 shows an example of a riding.

4 Analyzing Predictions

In this section we describe *Crystal*. One simple approach could be a system (see NGR system in Section 5) trained by a machine learning technique using n-gram features and classifying a message into multiple classes (e.g., NDP, Liberal, or Progressive). However, we develop a more sophisticated algorithm and compare its result with several baselines, including the simple n-gram method[2]. Experimental results in Section 5 show that *Crystal* outperforms all the baselines.

Our approach consists of three steps: feature generalization, classification using SVMs, and SVM result integration[3]. *Crystal* generates generalized sentences in the feature generalization step. Then it classifies each sentence using generalized lexical features in order to determine *Valence* of *Party* in a sentence. Finally, it combines results of sentences to determine *Valence* and *Party* of a message. Note that the classification using SVM is an intermediate step conducting a binary classification (i.e., WIN or LOSE) for the final multi-class classification in result integration. The following sections describe each step.

for each message M with a party that M predicts to win, P_w
for each sentence S_i in a message M
for each party P_j in S_i
valence V_j = +1 if P_j = P_w
valence V_j = -1 Otherwise
Generate S'_{ij} by substituting P_j with **PARTY**
and all other parties in S_i with **OTHER**
Return (P_j, V_j, S'_{ij})

Table 3. Feature generalization algorithm

4.1 Feature Generalization

In the feature generalization step, we generalize patterns of words used in predictive opinions. For example, instead of using three different trigrams like "Liberals will win", "NDP will win", and "Conservatives will win", we generalize these to "PARTY will win". The assumption is that the generalized patterns can represent better the relationship among Party, Valence, and words surrounding Party (e.g., will win) than pure lexical patterns. For this algorithm, we first substitute a candidate's name (both the first name and the last name) with the political party name that the candidate belongs to (see Table 2). We then break each message into sentences[4].

Table 3 outlines the feature generalization algorithm. Here, our approach is that if a message pre-

[2] N-gram approach is often unbeatable (and therefore great) in many text classification tasks.

[3] "feature" indicates n-grams in our corpus that we use in the SVM classification step.

[4] The sentence breaker that we used is available at http://search.cpan.org/ ~shlomoy/Lingua-EN-sentence - 0.25/lib/Lingua/EN/Sentence.pm.

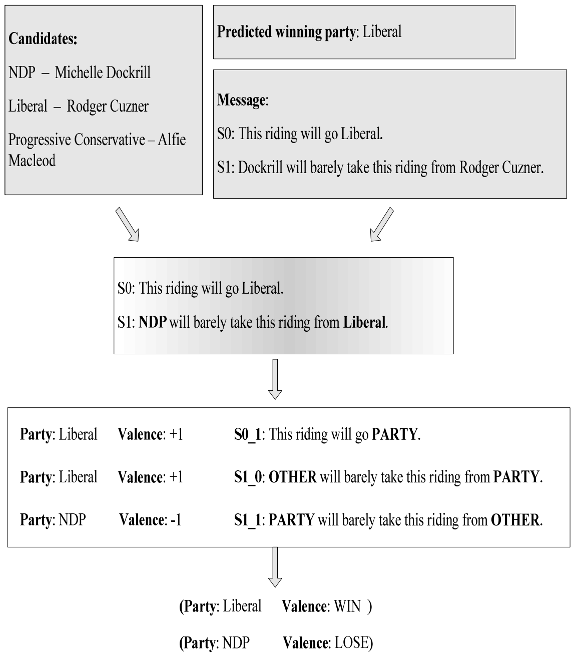

Figure 2. An example of feature generalization of a message

dicts a particular party to win, sentences which mention that party in the message also imply that it will win. Conversely all other parties are assumed to be in sentences that imply they will lose. As shown in Section 3.2, a message (*M*) in our corpus has a label of a party (P_w) that the author of *M* predicts to win. After breaking sentences in *M*, we duplicate a sentence by the number of unique parties in the sentence and modify the duplicated sentences by substituting the party names with PARTY and OTHER in order to generalize features.

Consider the following sentence:

"**Dockrill** will barely take this riding from **Rodger Cuzner**"

which gets re-written as:

"**NDP** will barely take this riding from **Liberal**"

because Dockrill is an NDP candidate and Rodger Cuzner is a Liberal candidate. Since the sentence contains two parties (i.e., NDP and Liberal), the algorithm duplicates the sentence twice, once for each party (see Lines 4–8 in Table 3)[5]. For NDP, the algorithm determines its Valence as -1 because NDP is not equal to the predicted winning party (i.e., Liberal) of the message (see Lines 4–5 in Table 3). Then it generates a generalized sentence by substituting NDP with PARTY and Liberal with OTHER (Lines 6–7). It returns (NDP, -1, "PARTY will barely take this riding from OTHER"). For Liberal, on the other hand, the algorithm determines its Valence as +1 since Liberal is the same as the predicted winning party of the message. After similar generalization, it returns (Liberal, +1, "OTHER will barely take this riding from PARTY").

Note that the final result of the feature generalization algorithm is a set of triplets: (Party, Valence, Generalized Sentence). Among a triplet, we use (Valence, Generalized Sentence) to produce feature vectors for a machine learning algorithm (see Section 4.2) and (Party, Valence) to integrate system results of each sentence for the final decision of Party and Valence of a message (see Section 4.3). Figure 2 shows an example of the algorithm.

4.2 Classification Using SVMs

In this step, we use Support Vector Machines (SVMs) to train our system using the generalized features described in Section 4.1. After we obtained examples of (Valence, Generalized Sentence) in the feature generalization step, we modeled a subtask of classifying a Generalized Sentence into Valence towards our final goal of determining (Valence, Party) of a message. This subtask is a binary classification since Valence has only 2 classes: +1 and -1[6]. Given a generalized sentence "OTHER will barely take this riding from PARTY" in Figure 2, for example, the goal of our system is to learn *WIN* valence for *PARTY*. Features for SVMs are extracted from generalized sentences. We implemented our SVM learning model using the SVM*light* package[7].

4.3 SVM Result Integration

In this step, we combine the valence of each sentence predicted by SVMs to determine the final valence and predicted party of a message. For each party mentioned in a message, we calculate the sum of the party's valences of each sentence and

[5] In the feature generalization algorithm, we represent WIN and LOSE valence as +1 and -1.

[6] However, the final evaluation of the system and all the baselines is equally performed on the multi-classification results of messages.

[7] SVM*light* is available from http://svmlight.joachims.org/

pick a party that has the maximum value. This integration algorithm can be represented as follows:

$$\arg\max_{p} \sum_{k=0}^{m} Valence_k(p)$$

where p is one of parties mentioned in a message, m is the number of sentences that contains party p in a message, and $Valence_k(p)$ is the valence of p in the k^{th} sentence that contains p. Given the example in Figure 2, the *Liberal* party appears twice in sentence S0 and S1 and its total valence score is +2, whereas the *NDP* party appears once in sentence S1 and its valence sum is -1. As a result, our algorithm picks liberal as the winning party that the message predicts.

5 Experiments and Results

This section reports our experimental results showing empirical evidence that *Crystal* outperforms several baseline systems.

5.1 Experimental Setup

Our corpus consists of 4858 and 4680 messages from 2004 and 2006 Canadian federal election prediction data respectively described in detail in Section 3.2. We split our pre-processed corpus into 10 folds for cross-validation. We implemented the following five systems to compare with *Crystal* [8].

- **NGR**: In this algorithm, we train the system using SVM with n-gram features without the generalization step described in Section 4.1 [9]. The replacement of each candidate's first and last name by his or her party name was still applied.
- **FRQ**: This system picks the most frequently mentioned party in a message as the predicted winning party. Party name substitution is also applied. For example, given a message "This riding will go liberal. Dockrill will barely take this riding from Rodger Cuzner.", all candidates' names are replaced by party names (i.e., "This riding will go Liberal. NDP will barely take this riding from Liberal."). After name replacement, the system picks Liberal as an answer because Liberal appears twice whereas NDP appears only once. Note that, unlike *Crystal*, this system does not consider the valence of each party (as done in our sentence duplication step of the feature generalization algorithm). Instead, it blindly picks the party that appeared most in a message.
- **MJR**: This system marks all messages with the most dominant predicted party in the entire data set. In our corpus, Conservatives was the majority party (3480 messages) followed closely by Liberal (3473 messages).
- **INC**: This system chooses the incumbent party as the predicted winning party of a message. (This is a strong baseline since incumbents often win in Canadian politics). For example, since the incumbent party of the riding "Blackstrap" in 2004 was Conservative, all the messages about Blackstrap in 2004 were marked Conservative as their predicted winning party by this system.
- **JDG**: This system uses judgment opinion words as its features for SVM. For our list of judgment opinion words, we use *General Inquirer* which is a publicly available list of 1635 positive and negative sentiment words (e.g., love, hate, wise, dumb, etc.)[10].

5.2 Experimental Results

We measure the system performance with its accuracy in two different ways: accuracy per message ($Acc_{message}$) and accuracy per riding (Acc_{riding}). Both accuracies are represented as follows:

$$Acc_{message} = \frac{\text{\# of messages the system correctly labled}}{\text{Total \# of messages in a test set}}$$

$$Acc_{riding} = \frac{\text{\# of ridings the system correctly predicted}}{\text{Total \# of ridings in a test set}}$$

We first report the results with $Acc_{message}$ in Evaluation1 and then report with Acc_{riding} in Evaluation2.

Evaluation1: Table 4 shows accuracies of baselines and *Crystal*. We calculated accuracy for each test set in 10-fold data sets and averaged it. Among the baselines, MJR performed worst (36.48%). Both FRQ and INC performed around 50% (54.82% and 53.29% respectively). NGR achieved its best score (62.02%) when using unigram, bigram, and trigram features together (uni+bi+tri). We also experimented with other feature combinations (see Table 5). Our system achieved 73.07% which is 11% higher than NGR and around 20%

[8] In our experiments using SVM, we used the linear kernel for all *Crystal*, NGR, and JDG.

[9] This system is exactly like *Crystal* without the feature generalization and result integration steps.

[10] Available at http://www.wjh.harvard.edu/~inquirer /homecat.htm

system	$Acc_{message}$ (%)	Acc_{riding} (%)
FRQ	54.82	63.14
MJR	36.48	36.63
INC	53.29	78.03
NGR (uni+bi+tri)	62.02	79.65
JDG	66.23	78.68
Crystal (uni+bi+tri)	**73.07**	**81.68**

Table 4. System performance with accuracy per message ($Acc_{message}$) and accuracy per riding (Acc_{riding}): FRQ, MJR, INC, NGR, JDG, and *Crystal*.

Features	$Acc_{message}$ (%)	
	NGR	*Crystal*
uni	60.49	72.03
bi	58.79	71.81
tri	54.04	69.57
four	47.25	67.64
uni + bi	61.54	72.93
uni + tri	61.36	72.20
uni + four	60.70	72.84
bi + tri	58.68	72.26
bi + four	58.54	72.17
uni + bi + tri	**62.02**	**73.07**
uni + bi + four	61.75	72.30
uni + tri + four	61.34	72.30
bi + tri + four	58.42	72.62
uni + bi + tri + four	61.96	73.01

Table 5. System performance with different features: Pure n-gram (NGR) and Generalized n-gram *Crystal.*

higher than FRQ and INC. The best accuracy of our system was also obtained with the combination of unigram, bigram, and trigram features.

The JDG system, which uses positive and negative sentiment word features, had 66.23% accuracy. This is about 7% lower than *Crystal*. Since the lower performance of JDG might be related to the number of features it uses, we also experimented with the reduced number of features of *Crystal* based on the *tfidf* scores[11]. With the same number of features (i.e., 1635), *Crystal* performed 70.62% which is 4.4% higher than JDG. An interesting finding was that NGR with 1635 features performed only 54.60% which is significantly

[11] The total number of all features of *Crystal* is 689,642.

Patterns in *WIN* class	Patterns in *LOSE* class
PARTY_will_win	want_OTHER
PARTY_hold	PARTY_don't_have
PARTY_will_win_this	OTHER_and
PARTY_win	the_PARTY
will_go_PARTY	OTHER_will_win
PARTY_will_take	OTHER_is
PARTY_will_take_this	to_the_OTHER
PARTY_is	and_OTHER
safest_PARTY	results_OTHER
PARTY_has	OTHER_has
go_PARTY_again	to_OTHER

Table 6. Examples of frequent features in WIN and LOSE classes.

lower than both systems. This indicates that the 1635 pure n-gram features are not as good as the same number of sentiment words carefully chosen from a dictionary but the generalized features of *Crystal* represent the predictive opinions better than JDG features.

Table 5 illustrates the comparison of NGR (without feature generalization) and *Crystal* (with feature generalization) in different feature combinations. *uni, bi, tri,* and *four* correspond to *unigram, bigram, trigram*, and *fourgram*. Our proposed technique *Crystal* performed always better than the pure n-gram system (NGR). Both systems performed best (62.02% and 73.07%) with the combination of unigram, bigram, and trigram (uni+bi+tri). The second best scores (61.96% and 73.01%) are achieved with the combinations of all grams (uni+bi+tri+four) in both systems. Using fourgrams alone performed worst since the system overfitted to the training examples.

Table 6 presents several examples of frequent n-gram features in both WIN and LOSE classes. As shown in Table 6, lexical patterns in the WIN class express optimistic sentiments about PARTY (e.g., PARTY_will_win and go_ PARTY_again) whereas patterns in the LOSE class express pessimistic sentiments (e.g., PARTY_don't_have) and optimistic ones about OTHER (e.g., want_OTHER).

Evaluation2: In this evaluation, we use Acc_{riding} computed as the number of ridings that a system correctly predicted, divided by the total number of ridings. For each riding *R*, systems pick a party that obtains the majority prediction votes from messages in *R* as the winning party of *R*. For example, if *Crystal* identified 9 messages predicting for Conservative Party, 3 messages for NDP, and 1 message for Liberal among 13 messages in the riding "Blackstrap", the system will predict that the Conservative Party would win in "Blackstrap".

Table 4 shows the system performance with Acc_{riding}. Note that people who write messages on a particular web site are not a random sample for prediction. So we introduce a measure of confidence (*ConfidenceScore*) of each system and use the prediction results when the *ConfidenceScore* is higher than a threshold. Otherwise, we use a default party (i.e., the incumbent party) as the winning party. *ConfidenceScore* of a riding *R* is calculated as follows:

$$ConfidenceScore = count_{message}(P_{first}) - count_{message}(P_{second})$$

where $count_{message}(P_x)$ is the number of messages that predict a party P_x to win, P_{first} is the party that the most number of messages predict to win, and P_{second} is the party that the second most number of messages predict to win.

We used 62 ridings to tune the *ConfidenceScore* parameter arriving at the value of 4. As shown in Table 4, the system which just considers the incumbent party (INC) performed fairly well (78.03% accuracy) because incumbents are often re-elected in Canadian elections. The upper bound of this prediction task is 88.85% accuracy which is the prediction result using numerical values of a prediction survey. FRQ and MJR performed 63.14% and 36.63% respectively. Similarly to Evaluation1, JDG which only uses judgment word features performed worse than both *Crystal* and NGR. Also, *Crystal* with our feature generalization algorithm performed better than NGR with non-generalized n-gram features. The accuracy of *Crystal* (81.68%) is comparable to the upper bound 88.85%.

6 Discussion

In this section, we discuss possible extensions and improvements of this work.

Our experiment focuses on investigating aspects of predictive opinions by learning lexical patterns and comparing them with judgment opinions. However, this work can be extended to investigating how those two types of opinions are related to each other and whether lexical features of one (e.g., judgment opinion) can help identify the other (e.g., predictive opinion). Combining two types of opinion features and testing on each domain can examine this issue.

In our experiment, we used General Inquirer words as judgment opinion indicators for JDG baseline system. It might be interesting to employ different resources for judgment words such as the polarity lexicon by Wilson et al. (2005) and the recently released SentiWordNet[12].

Our work is an initial step towards analyzing a new type of opinion. In the future, we plan to incorporate more features such as priors like incumbent party in addition to the lexical features to improve the system performance.

7 Conclusions

In this paper, we proposed a framework for working with *predictive opinion*. Previously, researchers in opinion analysis mostly focused on judgment opinions which express positive or negative sentiment about a topic, as in product reviews and policy discussions. Unlike judgment opinions, predictive opinions express a person's opinion about the future of a topic or event such as the housing market, a popular sports match, and election results, based on his or her belief and knowledge. Among these many kinds of predictive opinions, we focused on election prediction.

We collected past election prediction data from an election prediction project site and automatically built a gold standard. Using this data, we modeled the election prediction task using a supervised learning approach, SVM. We proposed a novel technique which generalized n-gram feature patterns. Experimental results showed that this approach outperforms several baselines as well as a non-generalized n-gram approach. This is significant because an n-gram model without generalization is often extremely competitive in many text classification tasks.

This work adopts NLP techniques for predictive opinions and it sets the foundation for exploring a whole new subclass of the opinion analysis problems. Potential applications of this work are systems that analyze various kinds of election predictions by monitoring texts in discussion boards and personal blogs. In the future, we would like to

[12] http://sentiwordnet.isti.cnr.it/

model predictive opinions in other domains such as the real estate market and the stock market which would require further exploration of system design and data collection.

Reference

Engelmore, R., and Morgan, A. eds. 1986. *Blackboard Systems*. Reading, Mass.: Addison-Wesley.

Dave, K., Lawrence, S. and Pennock, D. M. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. *Proc. of World Wide Web Conference 2003*

Drucker, H., Wu, D. and Vapnik, V. 1999. Support vector machines for spam categorization. *IEEE Trans*. Neural Netw., 10, pp 1048–1054.

Gliozzo, A., Strapparava C. and Dagan, I. 2005. Investigating Unsupervised Learning for Text Categorization Bootstrapping, *Proc. of EMNLP 2005*. Vancouver, B.C., Canada

Hu, M. and Liu, B. 2004. Mining and summarizing customer reviews. *Proc. Of KDD-2004*, Seattle, Washington, USA.

Jindal, N. and Liu, B. 2006. Mining Comprative Sentences and Relations. *Proc. of 21st National Conference on Artificial Intellgience (AAAI-2006).* 2006. Boston, Massachusetts, USA

Joachims, T. 1998. Text categorization with support vector machines: Learning with many relevant features, *Proc. of ECML*, p. 137–142.

Kim, S-M. and Hovy, E. 2004. Determining the Sentiment of Opinions. *Proc. of COLING 2004*.

Liu, B., Li, X., Lee, W. S. and Yu, P. S. Text Classification by Labeling Words *Proc. of AAAI-2004*, San Jose, USA.

Pang, B, Lee, L. and Vaithyanathan, S. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proc. of EMNLP 2002*.

Popescu, A-M. and Etzioni, O. 2005. Extracting Product Features and Opinions from Reviews, *Proc. of HLT-EMNLP 2005*.

Rickel, J. and Porter, B. 1997. Automated Modeling of Complex Systems to Answer Prediction Questions, *Artificial Intelligence Journal*, volume 93, numbers 1-2, pp. 201–260

Riloff, E., Wiebe, J., and Phillips, W. 2005. Exploiting Subjectivity Classification to Improve Information Extraction, *Proc. of the 20th National Conference on Artificial Intelligence (AAAI-05)* .

Riloff, E., Wiebe, J. and Wilson, T. 2003. Learning Subjective Nouns Using Extraction Pattern Bootstrapping. *Proc. of CoNLL 2003*. pp 25–32.

Rodionov, S. and Martin, J. H. 1996. A Knowledge-Based System for the Diagnosis and Prediction of Short-Term Climatic Changes in the North Atlantic, *Journal of Climate*, 9(8)

Turney, P. 2002. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Proc. of ACL 2002*, pp 417–424.

Wiebe, J., Bruce, R. and O'Hara, T. 1999. Development and use of a gold standard data set for subjectivity classifications. *Proc. of ACL 1999*, pp 246–253.

Wiebe, J., Wilson, T. , Bruce, R. , Bell , M. and Martin, M. Learning Subjective Language. 2004. *Computational Linguistics*

Wilson, T., Wiebe, J. and Hoffmann, P. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. *Proc. of HLT/EMNLP 2005*.

Yu, H. and Hatzivassiloglou, V. 2003. Towards Answering Opinion Questions: Separating Facts from Opinions and Identifying the Polarity of Opinion Sentences. *Proc. of EMNLP 2003*.