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