Adapting a Polarity Lexicon using Integer Linear Programming for Domain-Specific Sentiment Classification

Yejin Choi and Claire Cardie
Department of Computer Science
Cornell University
Ithaca, NY 14853
{ychoi,cardie}@cs.cornell.edu

Abstract

Polarity lexicons have been a valuable resource for sentiment analysis and opinion mining. There are a number of such lexical resources available, but it is often suboptimal to use them as is, because general purpose lexical resources do not reflect domain-specific lexical usage. In this paper, we propose a novel method based on integer linear programming that can adapt an existing lexicon into a new one to reflect the characteristics of the data more directly. In particular, our method collectively considers the relations among words and opinion expressions to derive the most likely polarity of each lexical item (*positive, neutral, negative*, or *negator*) for the given domain. Experimental results show that our lexicon adaptation technique improves the performance of fine-grained polarity classification.

1 Introduction

Polarity lexicons have been a valuable resource for sentiment analysis and opinion mining. In particular, they have been an essential ingredient for fine-grained sentiment analysis (e.g., Kim and Hovy (2004), Kennedy and Inkpen (2005), Wilson et al. (2005)). Even though the polarity lexicon plays an important role (Section 3.1), it has received relatively less attention in previous research. In most cases, polarity lexicon construction is discussed only briefly as a preprocessing step for a sentiment analysis task (e.g., Hu and Liu (2004), Moilanen and Pulman (2007)), but the effect of different alternative polarity lexicons is not explicitly investigated. Conversely, research efforts that focus on constructing a general purpose polarity lexicon (e.g., Takamura et al. (2005), Andreevskaia and Bergler (2006), Esuli and Sebastiani (2006), Rao and Ravichandran (2009)) generally evaluate the lexicon in isolation from any potentially relevant NLP task, and it is unclear how the new lexicon might affect end-to-end performance of a concrete NLP application.

It might even be unrealistic to expect that there can be a general-purpose lexical resource that can be effective across *all* relevant NLP applications, as general-purpose lexicons will not reflect domain-specific lexical usage. Indeed, Blitzer et al. (2007) note that the polarity of a particular word can carry opposite sentiment depending on the domain (e.g., Andreevskaia and Bergler (2008)).

In this paper, we propose a novel method based on integer linear programming to adapt an existing polarity lexicon into a new one to reflect the characteristics of the data more directly. In particular, our method considers the relations among words and opinion expressions collectively to derive the most likely polarity of each word for the given domain.

Figure 1 depicts the key insight of our approach using a bipartite graph. On the left hand side, each node represents a word, and on the right hand side, each node represents an opinion expression. There is an edge between a word w_i and an opinion expression e_j, if the word w_i appears in the expression e_j. We assume the possible polarity of each expression is one of the following three values: {*positive, neutral, negative*}, while the possible polarity of each word is one of: {*positive*, *neutral*, *negative* or *negator*}. Strictly speaking, *negator* is not a value for polarity, but we include them in our lexicon, because *valence shifters* or *negators* have been shown to play an important role for sentiment analysis (e.g., Polanyi and Zaenen (2004), Moilanen and Pulman (2007), Choi and Cardie (2008)).

Typically, the ultimate goal of the sentiment analysis task is to determine the expression-level (or sentiment/ document-level) polarities, rather

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 590–598,
Singapore, 6-7 August 2009.

than the correct word-level polarities with respect to the domain. Therefore, word-level polarities can be considered as latent information. In this paper, we show how we can improve the word-level polarities of a general-purpose polarity lexicon by utilizing the expression-level polarities, and in return, how the adapted word-level polarities can improve the expression-level polarities.

In Figure 1, there are two types of relations we could exploit when adapting a general-purpose polarity lexicon into a domain-specific one. The first are word-to-word relations within each expression. That is, if we are not sure about the polarity of a certain word, we can still make a guess based on the polarities of other words within the same expression and knowledge of the polarity of the expression. The second type of relations are word-to-expression relations: e.g., some words appear in expressions that take on a variety of polarities, while other words are associated with expressions of one polarity class or another.

In relation to previous research, analyzing word-to-word (intra-expression) relations is most related to techniques that determine expression-level polarity *in context* (e.g., Wilson et al. (2005)), while exploring word-to-expression (inter-expression) relations has connections to techniques that employ more of a global-view of corpus statistics (e.g., Kanayama and Nasukawa (2006)).[1]

While most previous research exploits only one or the other type of relation, we propose a unified method that can exploit both types of semantic relation, while adapting a general purpose polarity lexicon into a domain specific one. We formulate our lexicon adaptation task using integer linear programming (ILP), which has been shown to be very effective when solving problems with complex constraints (e.g., Roth and Yih (2004), Denis and Baldridge (2007)). And the word-to-word and word-to-expression relations discussed above can be encoded as soft and hard constraints in ILP. Unfortunately, one class of constraint that we would like to encode (see Section 2) will require an exponentially many number of constraints when grounded into an actual ILP problem. We therefore propose an approximation scheme to make the problem more practically solvable.

We evaluate the effect of the adapted lexicon in the context of a concrete NLP task: expression-level polarity classification. Experimental results show that our lexicon adaptation technique improves the accuracy of two competitive expression-level polarity classifiers from 64.2% - 70.4% to 67.0% - 71.2%..

[1] In case of document-level polarity classification, word-to-expression relations correspond to word-to-document relations.

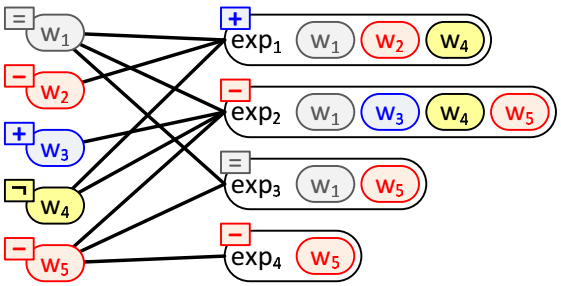

Figure 1: The relations among words and expressions. + indicates positive, - indicates negative, = indicates neutral, and ¬ indicates a negator.

2 An Integer Linear Programming Approach

In this section, we describe how we formulate the lexicon adaptation task using integer linear programming. Before we begin, we assume that we have a general-purpose polarity lexicon $\mathcal{L}$, and a polarity classification algorithm $f(e_l, \mathcal{L})$, that can determine the polarity of the opinion expression e_l based on the words in e_l and the initial lexicon $\mathcal{L}$. The polarity classification algorithm $f(\cdot)$ can be either a heuristic-based one, or a machine-learning based one – we consider it as a black box for now.

Constraints for word-level polarities: For each word x_i, we define four binary variables: $x_i^+, x_i^=, x_i^-, x_i^\neg$ to represent positive, neutral, negative polarity, and negators respectively. If $x_i^\delta = 1$ for some $\delta \in \{+, =, -, \neg\}$, then the word x_i has the polarity δ. The following inequality constraint states that at least one polarity value must be chosen for each word.

$$x_i^+ + x_i^= + x_i^- + x_i^\neg >= 1 \qquad (1)$$

If we allow only one polarity per word, then the above inequality constraint should be modified as an equality constraint. Although most words tend to associate with a single polarity, some can take on more than one polarity. In order to capture this observation, we introduce an auxiliary binary variable α_i for each word x_i. Then the next inequality

constraint states that at most two polarities can be chosen for each word.

$$x_i^+ + x_i^= + x_i^- + x_i^\neg <= 1 + \alpha_i \quad (2)$$

Next we introduce the initial part of our objective function.

$$\text{maximize} \sum_i \Big(w_i^+ x_i^+ + w_i^= x_i^= + w_i^- x_i^- + w_i^\neg x_i^\neg - w_\alpha \alpha_i \Big) + \cdots \quad (3)$$

For the auxiliary variable α_i, we apply a constant weight w_α to discourage ILP from choosing more than one polarity for each word. We can allow more than two polarities for each word, by adding extra auxiliary variables and weights. For each variable x_i^δ, we define its weight w_i^δ, which indicates how likely it is that word x_i carries the polarity δ. We define the value of w_i^δ using two different types of information as follows:

$$w_i^\delta := {}^{\mathcal{L}}w_i^\delta + {}^{\mathcal{C}}w_i^\delta$$

where ${}^{\mathcal{L}}w_i^\delta$ is the degree of polarity δ for word x_i determined by the general-purpose polarity lexicon $\mathcal{L}$, and ${}^{\mathcal{C}}w_i^\delta$ is the degree of polarity δ determined by the corpus statistics as follows:[2]

$${}^{\mathcal{C}}w_i^\delta := \frac{\#\text{ of } x_i \text{ in expressions with polarity } \delta}{\#\text{ of } x_i \text{ in the corpus } \mathcal{C}}$$

Note that the occurrence of word x_i in an expression e_j with a polarity δ does not necessarily mean that the polarity of x_i should also be δ, as the interpretation of the polarity of an expression is more than just a linear sum of the word-level polarities (e.g., Moilanen and Pulman (2007)). Nonetheless, not all expressions require a complicated inference procedure to determine their polarity. Therefore, ${}^{\mathcal{C}}w_i^\delta$ still provides useful information about the likely polarity of each word based on the corpus statistics.

From the perspective of Chomskyan linguistics, the weights ${}^{\mathcal{L}}w_i^\delta$ based on the prior polarity from the lexicon can be considered as having a "competence" component , while ${}^{\mathcal{C}}w_i^\delta$ derived from the corpus counts can be considered as a "performance" component (Noam Chomsky (1965)).

[2]If a word x_i is in an expression that is not an opinion, then we count it as an occurrence with neutral polarity.

Constraints for content-word negators: Next we describe a constraint that exploits knowledge of the typical distribution of *content-word negators* in natural language. Content-word negators are words that are not function words, but act semantically as negators (Choi and Cardie, 2008).[3] Although it is possible to artificially construct a very convoluted sentence with lots of negations, it is unlikely for multiple layers of negations to appear very often in natural language (Pickett et al. (1996)). Therefore, we allow at most one content-word negator for each expression e_l. Because we do not restrict the number of function-word negators, our constraint still gives room for multiple layers of negations.

$$\sum_{i \in \mu(e_l)} x_i^\neg <= 1 \quad (4)$$

In the above constraint, $\mu(e_l)$ indicates the set of indices of content words appearing in e_l. For instance, if $i \in \mu(e_l)$, then x_i appears in e_l. This constraint can be polished further to accommodate longer expressions where multiple content-word negators are more likely to appear, by adding a separate constraint with a sliding window.

Constraints for expression-level polarities: Before we begin, we introduce $\pi(e_l)$ that will be used often in the remaining section. For each expression e_l, we define $\pi(e_l)$ to be the set of content words appearing in e_l, together with the most likely polarity proposed by a general-purpose polarity lexicon $\mathcal{L}$. For instance, if $x_i^+ \in \pi(e_l)$, then the polarity of word x_i is $+$ according to $\mathcal{L}$.

Next we encode constraints that consider expression-level polarities. If the polarity classification algorithm $f(e_l, \mathcal{L})$ makes an incorrect prediction for e_l using the original lexicon $\mathcal{L}$, then we need to encourage ILP to fix the error by suggesting different word-level polarities. We capture this idea by the following constraint:

$$\sum_{x_i^\delta \in \pi(e_l)} x_i^\delta <= |\pi(e_l)| - 1 + \beta_l \quad (5)$$

The auxiliary binary variable β_l is introduced for each e_l so that the assignment $\pi(e_l)$ does not have to be changed if paying for the cost w_β in the objective function. (See equation (10).) That is, suppose the ILP solver assigns '1' to all variables

[3]Examples of content-word negators are *destroy*, *eliminate*, *prevent* etc.

in $\phi(e_l)$, (which corresponds to keeping the original lexicon as it is for all words in the given expression e_l), then the auxiliary variable β_l must be also set as '1' in order to satisfy the constraint (5). Because β_l is associated with a negative weight in the objective function, doing so will act against maximizing the objective function. This way, we discourage the ILP solver to preserve the original lexicon as it is.

To verify the constraint (5) further, suppose that the ILP solver assigns '1' for all variables in $\phi(e_l)$ except for one variable. (Notice that doing so corresponds to proposing a new polarity for one of the words in the given expression e_l.) Then the constraint (5) will hold regardless of whether the ILP solver assigns '0' or '1' to β_l. Because β_l is associated with a negative weight in the objective function, the ILP solver will then assign '0' to β_l to maximize the objective function. In other words, we encourage the ILP solver to modify the original lexicon for the given expression e_l.

We use this type of *soft* constraint in order to cope with the following two noise factors: first, it is possible that some annotations are noisy. Second, $f(e_l, \mathcal{L})$ is not perfect, and might not be able to make a correct prediction even with the correct word-level polarities.

Next we encode a constraint that is the opposite of the previous one. That is, if the polarity classification algorithm $f(e_l, \mathcal{L})$ makes a correct prediction on e_l using the original lexicon $\mathcal{L}$, then we encourage ILP to keep the original word-level polarities for words in e_l.

$$\sum_{x_i^\delta \in \pi(e_l)} x_i^\delta \ >= \ |\pi(e_l)| - |\pi(e_l)|\beta_l \qquad (6)$$

Interpretation of constraint (6) with the auxiliary binary variable β_l is similar to that of constraint (5) elaborated above.

Notice that in equation (5), we encouraged ILP to fix the current lexicon $\mathcal{L}$ for words in e_l, but we have not specified the consequence of a modified lexicon ($\mathcal{L}'$) in terms of expression-level polarity classification $f(e_l, \mathcal{L}')$. Certain changes to $\mathcal{L}$ might not fix the prediction error for e_l, and those might even cause extra incorrect predictions for other expressions. Then it would seem that we need to replicate constraints (5) & (6) for all permutations of word-level polarities. However, doing so would incur exponentially many number of constraints ($4^{|e_l|}$) for each expression.[4]

To make the problem more practically solvable, we only consider changes to the lexicon that are within edit-one distance with respect to $\pi(e_l)$. More formally, let us define $\pi'(e_l)$ to be the set of content words appearing in e_l, together with the most likely polarity proposed by a modified polarity lexicon $\mathcal{L}'$. Then we need to consider all $\pi'(e_l)$ such that $|\pi'(e_l) \cap \pi(e_l)| = |\pi(e_l)| - 1$. There are $(4-1)|e_l|$ number of different $\pi'(e_l)$, and we index them as $\pi'_k(e_l)$. We then add following constraints similarly as equation (5) & (6):

$$\sum_{x_i^\delta \in \pi'_k(e_l)} x_i^\delta \ <= \ |\pi'_k(e_l)| - 1 + \beta_{(l,k)} \qquad (7)$$

if the polarity classification algorithm $f(\cdot)$ makes an incorrect prediction based on $\pi'_k(e_l)$. And,

$$\sum_{x_i^\delta \in \pi'_k(e_l)} x_i^\delta \ >= \ |\pi'_k(e_l)| - |\pi'_k(e_l)|\beta_{(l,k)} \qquad (8)$$

if the polarity classification algorithm $f(\cdot)$ makes a correct prediction based on $\pi'_k(e_l)$. Remember that none of the constraints (5) - (8) enforces assignment $\pi(e_l)$ or $\pi'_k(e_l)$ as a hard constraint. In order to enforce at least one of them to be chosen, we add the following constraint:

$$\sum_{x_i^\delta \in \pi(e_l)} x_i^\delta \ >= \ |\pi(e_l)| - 1 \qquad (9)$$

This constraint ensures that the modified lexicon $\mathcal{L}'$ is not drastically different from $\mathcal{L}$. Assuming that the initial lexicon $\mathcal{L}$ is a reasonably good one, constraining the search space for $\mathcal{L}'$ will regulate that $\mathcal{L}'$ does not turn into a degenerative one that overfits to the current corpus $\mathcal{C}$.

Objective function: Finally, we introduce our full objective function.

[4]For certain simple polarity classification algorithm $f(e_l, \mathcal{L})$, it is possible to write polynomially many number of constraints. However our approach intends to be more general by treating $f(e_l, \mathcal{L})$ as a black box, so that algorithms that do not factor nicely can also be considered as an option.

$$\begin{aligned}\text{maximize} \sum_i \Big(& w_i^+ x_i^+ + w_i^= x_i^= \\ + & w_i^- x_i^- + w_i^\neg x_i^\neg \\ - & w_\alpha \alpha_i \Big) \\ -\sum_l & w_\beta \rho_l \beta_l \\ -\sum_{l,k} & w_\beta \rho_{(l,k)} \beta_{(l,k)} \end{aligned} \quad (10)$$

We have already described the first part of the objective function (equation (3)), thus we only describe the last two terms here. w_β is defined similarly as w_α; it is a constant weight that applies for any auxiliary binary variable β_l and $\beta_{(l,k)}$.

We further define ρ_l and $\rho_{(l,k)}$ as secondary weights, or *amplifiers* to adjust the constant weight w_β. To enlighten the motivation behind the amplifiers ρ_l and $\rho_{(l,k)}$, we bring out the following observations:

1. Among the incorrect predictions for expression-level polarity classification, some are more incorrect than the other. For instance, classifying positive class to negative class is more wrong than classifying positive class to neutral class. Therefore, the cost of not fixing very incorrect predictions should be higher than the cost of not fixing less incorrect predictions. (See [R2] and [R3] in Table 1.)

2. If the current assignment $\pi(e_l)$ for expression e_l yields a correct prediction using the classifier $y(e_l, \mathcal{L})$, then there is not much point in changing $\mathcal{L}$ to $\mathcal{L}'$, even if $y(e_l, \mathcal{L}')$ also yields a correct prediction. In this case, we would like to assign slightly higher confidence in the original lexicon $\mathcal{L}$ then the new one $\mathcal{L}'$. (See [R1] in Table 1.)

3. Likewise, if the current assignment $\pi(e_l)$ for expression e_l yields an incorrect prediction using the classifier $y(e_l, \mathcal{L})$, then there is not much point in changing $\mathcal{L}$ to $\mathcal{L}'$, if $y(e_l, \mathcal{L}')$ also yields an equally incorrect prediction. Again we assign slightly higher confidence in the original lexicon $\mathcal{L}$ than the new one $\mathcal{L}'$ in such cases. (Compare each row in [R2] with a corresponding row in [R3] in Table 1.)

[R1]	If $\pi(e_l)$ correct	ρ_l	$\leftarrow 1.5$
	If $\pi'_k(e_l)$ correct	$\rho_{(l,k)}$	$\leftarrow 1.0$
[R2]	If $\pi(e_l)$ very incorrect	ρ_l	$\leftarrow 1.0$
	If $\pi(e_l)$ less incorrect	ρ_l	$\leftarrow 0.5$
[R3]	If $\pi'_k(e_l)$ very incorrect	$\rho_{(l,k)}$	$\leftarrow 1.5$
	If $\pi'_k(e_l)$ less incorrect	$\rho_{(l,k)}$	$\leftarrow 1.0$

Table 1: The value of amplifiers ρ_l and $\rho_{(l,k)}$.

To summarize, for correct predictions, the degree of ρ determines the degree of cost of (undesirably) altering the current lexicon for e_l. For incorrect predictions, the degree of ρ determines the degree of cost of not fixing the current lexicon for e_l.

3 Experiments

In the experiment section, we seek for answers for the following questions:

Q1 What is the effect of a polarity lexicon on the expression-level polarity classification task? In particular, is it useful when using a machine learning technique that might be able to learn the necessary polarity information just based on the words in the training data, without consulting a dictionary? (Section 3.1)

Q2 What is the effect of an adapted polarity lexicon on the expression-level polarity classification task? (Section 3.2)

Notice that we include the *neutral* polarity in the polarity classification. It makes our task much harder (e.g., Wilson et al. (2009)) than those that assume inputs are guaranteed to be either strongly positive or negative (e.g., Pang et al. (2002), Choi and Cardie (2008)). But in practice, one cannot expect that a given input is strongly polar, as automatically extracted opinions are bound to be noisy. Furthermore, Wiebe et al. (2005) discuss that some opinion expressions do carry a neutral polarity.

We experiment with the Multi-Perspective Question Answering (MPQA) corpus (Wiebe et al., 2005) for evaluation. It contains 535 newswire documents annotated with phrase-level subjectivity information. We evaluate on all opinion expressions that are known to have high level of inter-annotator agreement. That is, we include opinions with intensity marked as 'medium' or

higher, and exclude those with annotation confidence marked as 'uncertain'. To focus our study on the direct influence of the polarity lexicon upon the sentiment classification task, we assume the boundaries of the expressions are given. However, our approach can be readily used in tandem with a system that extracts opinion expressions (e.g., Kim and Hovy (2005), Breck et al. (2007)). Performance is reported using 10-fold cross-validation on 400 documents, and a separate 135 documents were used as a development set. For the general-purpose polarity lexicon, we expand the polarity lexicon of Wilson et al. (2005) with General Inquirer dictionary as suggested by Choi and Cardie (2008).

We report the performance in two measures: *accuracy* for 3-way classification, and *average error distance*. The reason why we consider *average error distance* is because classifying a positive class into a negative class is worse than classifying a positive class into a neutral one. We define the error distance between 'neutral' class and any other class as 1, while the error distance between 'positive' class and 'negative' class as 2. If a predicted polarity is correct, then the error distance is 0. We compute the error distance of each prediction and take the average over all predictions in the test data.

3.1 Experiment-I: Effect of a Polarity Lexicon

To verify the effect of a polarity lexicon on the expression-level polarity classification task, we experiment with simple classification-based machine learning technique. We use the Mallet (McCallum, 2002) implementation of Conditional Random Fields (CRFs) (Lafferty et al., 2001).[5] To highlight the influence of a polarity lexicon, we compare the performance of CRFs with and without features derived from polarity lexicons.

Features: We encode basic features as words and lemmas for all content words in the given expression. The performance of CRFs using only the basic features are given in the first row of the Table 2. Next we encode features derived from polarity lexicons as follows.

- The output of Vote & Flip algorithm. (Section 3.2 & Figure 2.)

[5]We use the CRF implementation of Mallet (McCallum, 2002) with Markov-order 0, which is equivalent to Maximum Entropy models (Berger et al. (1996)).

	Accuracy	Avg. Error Distance
Without Lexicon	63.9	0.440
With Lexicon	70.4	0.334

Table 2: Effect of a polarity lexicon on expression-level classification using CRFs

- Number of positive, neutral, negative, and negators in the given expression.
- Number of positive (or negative) words in conjunction with number of negators.
- (boolean) Whether the number of positive words dominates negative ones.
- (boolean) Whether the number of negative words dominates positive ones.
- (boolean) None of the above two cases
- Each of the above three boolean values in conjunction with the number of negators.

Results: Table 2 shows the performance of CRFs with and without features that consult the general-purpose lexicon. As expected, CRFs can perform reasonably well (accuracy = 63.9%) even without consulting the dictionary, by learning directly from the data. However, having the polarity lexicon boosts the performance significantly (accuracy = 70.4%), demonstrating that lexical resources are very helpful for fine-grained sentiment analysis. The difference in performance is statistically significant by paired t-test for both accuracy ($p < 0.01$) and average error distance ($p < 0.01$).

3.2 Experiment-II: Adapting a Polarity Lexicon

In this section, we assess the quality of the adapted lexicon in the context of an expression-level polarity classification task. In order to perform the lexicon adaptation via ILP, we need an expression-level polarity classification algorithm $f(e_l, \mathcal{L})$ as described in Section 2. According to Choi and Cardie (2008), voting algorithms that recognize content-word negators achieve a competitive performance, so we will use a variant of it for simplicity. Because none of the algorithms proposed by Choi and Cardie (2008) is designed to handle the neutral polarity, we invent our own version as shown in Figure 2.

For each expression e_i,
  $nPositive \leftarrow$ # of positive words in e_i
  $nNeutral \leftarrow$ # of neutral words in e_i
  $nNegative \leftarrow$ # of negative words in e_i
  $nNegator \leftarrow$ # of negating words in e_i

  if ($nNegator$ % 2 = 0)
    then $fFlipPolarity \leftarrow false$
  else
    then $fFlipPolarity \leftarrow true$

  if ($nPositive > nNegative$) & $\neg$ $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $positive$
else if ($nPositive > nNegative$) & $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $negative$
else if ($nPositive < nNegative$) & $\neg$ $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $negative$
else if ($nPositive < nNegative$) & $fFlipPolarity$
    then Polarity(e_i) $\leftarrow$ $neutral$
else if $nNeutral > 0$
    then Polarity(e_i) $\leftarrow$ $neutral$
else
    then Polarity(e_i) $\leftarrow$ $default\_polarity$ (the most prominent polarity in the corpus)

Figure 2: Vote & Flip Algorithm

It might look a bit complex at first glance, but the intuition is simple. The variable $fFlipPolarity$ determines whether we need to flip the overall majority polarity based on the number of negators in the given expression. If the positive (or negative) polarity words dominate the given expression, and if there is no need to flip the majority polarity, then we take the positive (or negative) polarity as the overall polarity. If the positive (or negative) polarity words dominate the given expression, and if we need to flip the majority polarity, then we take the negative (or neutral) polarity as the overall polarity.

Notice that the result of flipping the negative polarity is *neutral*, not *positive*. In our pilot study, we found that this strategy works better than flipping the negative polarity to positive.[6] Finally, if the number of positive words and the negative words tie, and there is any neutral word, then we assign the neutral polarity. In this case, we don't worry if there is a negator, because flipping a neutral polarity would still result in a neutral polarity. If none of above condition is met, than we default to the most prominent polarity of the data, which is the negative polarity in the MPQA corpus. We name this simple algorithm as Vote & Flip algorithm. The performance is shown in the first row in Table 2.

Next we describe the implementation part of the ILP. For 10 fold-cross validation, we formulate the ILP problem using the training data (360 documents), and then test the effect of the adapted lexicon on the remaining 40 documents. We include only those content words that appeared more than 3 times in the training data. From the pilot test using the development set, we picked the value of w_β as 0.1. We found that having the auxiliary variables α_l which allow more than one polarity per word does not necessarily help with the performance, so we omitted them. We suspect it is because the polarity classifiers we experimented with is not highly capable of disambiguating different lexical usages and select the right polarity for a given context. We use CPLEX integer programming solver to solve our ILP problems. On a machine with 4GHz CPU, it took several minutes to solve each ILP problem.

In order to assess the effect of the adapted lexicon using CRFs, we need to first train the CRFs model. Using the same training set used for the lexicon adaptation would be suboptimal, because the features generated from the adapted lexicon will be unrealistically good in that particular data. Therefore, we prepared a separate training data for CRFs using 135 documents from the development set.

Results: Table 3 shows the comparison of the original lexicon and the adapted lexicon in terms of polarity classification performance using the Vote & Flip algorithm. The adapted lexicon improves the accuracy as well as reducing the average error distance. The difference in performance is statistically significant by paired t-test for both accuracy ($p < 0.01$) and average error distance ($p < 0.01$).

Table 4 shows the comparison of the original lexicon and the adapted lexicon using CRFs. The improvement is not as substantial as that of Vote & Flip algorithm but the difference in performance is also statistically significant for both accuracy ($p = 0.03$) and average error distance ($p = 0.04$).

[6]This finding is not surprising. For instance, if we consider the polarity of *"She did not get hurt much from the accident."*, it can be viewed as neutral; although it is good that one did not hurt much, it is still bad that there was an accident. Hence it gives a mixed feeling, which corresponds to the neutral polarity.

	Accuracy	Avg. Error Distance
Original Lexicon	64.2	0.395
Adapted Lexicon	67.0	0.365

Table 3: Effect of an adapted polarity lexicon on expression-level classification using the Vote & Flip Algorithm

	Accuracy	Avg. Error Distance
Original Lexicon	70.4	0.334
Adapted Lexicon	71.2	0.327

Table 4: Effect of an adapted polarity lexicon on expression-level classification using CRFs

4 Related Work

There are a number of previous work that focus on building polarity lexicons (e.g., Takamura et al. (2005), Kaji and Kitsuregawa (2007), Rao and Ravichandran (2009)). But most of them evaluated their lexicon in isolation from any potentially relevant NLP task, and it is unclear how the new lexicon might affect end-to-end performance of a concrete NLP application. Our work differs in that we try to draw a bridge between general purpose lexical resources and a domain-specific NLP application.

Kim and Hovy (2005) and Banea et al. (2008) present bootstrapping methods to construct a subjectivity lexicon and measure the effect of the new lexicon for sentence-level subjectivity classification. However, their lexicons only tell whether a word is a subjective one, but not the polarity of the sentiment. Furthermore, the construction of lexicon is still an isolated step from the classification task. Our work on the other hand allows the classification task to directly influence the construction of lexicon, enabling the lexicon to be adapted for a concrete NLP application and for a specific domain.

Wilson et al. (2005) pioneered the expression-level polarity classification task using the MPQA corpus. The experimental results are not directly comparable to ours, because Wilson et al. (2005) limit the evaluation only for the words that appeared in their polarity lexicon. Choi and Cardie (2008) also focus on the expression-level polarity classification, but their evaluation setting is not as practical as ours in that they assume the inputs are guaranteed to be either strongly positive or negative.

5 Conclusion

In this paper, we present a novel lexicon adaptation technique based on integer linear programming to reflect the characteristics of the domain more directly. In particular, our method collectively considers the relations among words and opinion expressions to derive the most likely polarity of each lexical item for the given domain. We evaluate the effect of our lexicon adaptation technique in the context of a concrete NLP application: expression-level polarity classification. The positive results from our experiments encourage further research for lexical resource adaptation techniques.

Acknowledgments

This work was supported in part by National Science Foundation Grant BCS-0624277 and by the Department of Homeland Security under ONR Grant N0014-07-1-0152. We also thank the EMNLP reviewers for insightful comments.

References

Alina Andreevskaia and Sabine Bergler. 2008. When Specialists and Generalists Work Together: Overcoming Domain Dependence in Sentiment Tagging. *ACL*

Alina Andreevskaia and Sabine Bergler. 2006. Mining WordNet For a Fuzzy Sentiment: Sentiment Tag Extraction From WordNet Glosses. *EACL*

Carmen Banea, Rada Mihalcea, and JanyceWiebe. 2008. A Bootstrapping Method for Building Subjectivity Lexicons for Languages with Scarce Resources. *LREC*

Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. 1996. A maximum entropy approach to natural language processing. In *Computational Linguistics, 22(1)*

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, Boom-boxes, and Blenders: Domain Adaptation for Sentiment Classification. *Association for Computational Linguistics - ACL 2007*

Eric Breck, Yejin Choi and Claire Cardie. 2007. Identifying Expressions of Opinion in Context. In *IJCAI*.

Yejin Choi and Claire Cardie. 2008. Learning with Compositional Semantics as Structural Inference for Subsentential Sentiment Analysis. *EMNLP*

Noam Chomsky. 1965. Aspects of the theory of syntax. Cambridge, MA: MIT Press.

Pascal Denis and Jason Baldridge. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. *NAACL*

Andrea Esuli and Fabrizio Sebastiani. 2006. SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. In *Proceedings of 5th Conference on Language Resources and Evaluation (LREC),*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004)*.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents. In EMNLP-CoNLL.

Hiroshi Kanayama Tetsuya Nasukawa. 2006. Fully Automatic Lexicon Expansion for Domain-oriented Sentiment Analysis. In ACL.

Alistair Kennedy and Diana Inkpen. 2005. Sentiment Classification of Movie and Product Reviews Using Contextual Valence Shifters. In *Proceedings of FINEXIN 2005, Workshop on the Analysis of Informal and Formal Information Exchange during Negotiations.*

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING*.

Soo-Min Kim and Eduard Hovy. 2005. Automatic Detection of Opinion Bearing Words and Sentences. In Companion Volume to the *Proceedings of the Second International Joint Conference on Natural Language Processing (IJCNLP-05)*

John Lafferty, Andrew Kachites McCallum and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. http://mallet.cs.umass.edu.

Karo Moilanen and Stephen Pulman. 2007. Sentiment Composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007).*

Bo Pang, Lillian Lee and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *EMNLP*.

Joseph Pickett et al. 1996. The American heritage book of English usage: A practical and authoritative guide to contemporary English. Houghton Mifflin Company.

Livia Polanyi and Annie Zaenen. 2004. Contextual lexical valence shifters. In *Exploring Attitude and Affect in Text: Theories and Applications: Papers from the 2004 Spring Symposium, AAAI*.

Delip Rao and Deepak Ravichandran. 2009. Semi-Supervised Polarity Lexicon Induction. In *EACL*.

Dan Roth and Wen-tau Yih. 2004. A Linear Programming Formulation for Global Inference in Natural Language Tasks. In *CoNLL*.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *ACL*.

Janyce Wiebe, Theresa Wilson and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. In *Language Resources and Evaluation (formerly Computers and the Humanities), 39(2-3):165210*.

Theresa Wilson, Janyce Wiebe and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of HLT/EMNLP.*

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2009. Recognizing Contextual Polarity: an exploration of features for phrase-level sentiment analysis. In *Computational Linguistics 35(3).*