Targeted Sentiment Analysis: Identifying Student Sentiment Toward Courses and Instructors

Charles Welch and Rada Mihalcea

University of Michigan, Ann Arbor, MI
{cfwelch,mihalcea}@umich.edu

Abstract. We examine targeted sentiment analysis for the purpose of building a dialog system for academic advising. For dialog tasks such as course selection it is important to recognize the entities (i.e., courses and instructors) and the sentiment that students express toward them. We examine the effect of domain specific features, and show performance improvements for both the entity recognition and sentiment analysis tasks over baseline methods. A discussion of errors provides insight into the limitations of our method and directions for future work.

1 Introduction

Sentiment analysis is the computational study of people's opinions or emotions toward entities and their aspects [12], be those persons, organizations, or objects that a person expresses an opinion toward. It is a challenging problem that is increasingly being used for decision making by individuals and organizations [7].

The goal of the work described in this paper is to build a system that can automatically identify the sentiment that students hold towards courses and instructors. While most of the work to date in sentiment analysis has addressed the task of identifying the presence of sentiment clues inside a document, sentence, or phrase, in this paper we address the problem of "targeted sentiment," which aims to identify the sentiment that the speaker of a statement (i.e., a student) has toward selected entities (i.e., courses and instructors). We therefore perform a dual task: (1) entity extraction, where courses and instructors are automatically identified in text; followed by (2) sentiment analysis, where the sentiment held by the speaker toward those entities is also identified.

The work described in this paper is part of a larger project that aims to create a dialog system to provide academic advising support to undergraduate students. The language used in academic advising sessions is naturally complex, as advising generally covers a wide range of topics, ranging from course selection, to degree requirements, to academic challenges and exception requests. At this stage, the project is specifically targeting the task of course selection, and thus the current goal is to construct a dialog system that can assist students in planning what classes to take in their next semester. Correspondingly, the natural language understanding component is focused on understanding the student utterances during the dialog, and specifically on understanding what classes and instructors they talk about, and what their sentiment is (if any) towards these entities. Intuitively this is an important task because we already know many things about the student from their information stored in university databases. We do not, however, have information about their feelings toward their experiences, which are important to know when making recommendations about what classes they should take in the future.

In Section 2 we describe how this work fits into the bigger picture of sentiment analysis research and what has been done before. Section 3 describes the data that was used for these experiments, how it was collected, and statistics about its contents. Section 4 shows how entities are extracted from text for use in sentiment analysis and Section 5 describes how the sentiment toward entities is classified using these extracted entities or the ground truth. The paper ends with a discussion of results in Section 6 and conclusions in Section 7.

2 Background

Most work in sentiment analysis is done at one of three levels, document level, sentence level, and aspect level. In general, an opinion can be represented by the following quintuple, $(e_i, a_{ij}, oo_{ijkl}, h_k, t_l)$ [12]. The value e_i here represents the ith entity and a_{ij} represents aspect j of this entity. The kth holder of the opinion, is represented by h_k and the time, l, that the opinion is expressed is given by t_l. Given the entity, aspect, holder, and time, one can reason about an opinion orientation oo_{ijkl}. This is usually a positive, negative, or neutral value but in some work includes a larger number of possible orientations.

Sentiment analysis tasks can be categorized by which variables of this tuple are of interest. Given the intended application of a dialog system, h_k should be known, as the system speaks to individual students that have authenticated themselves to the system. The time, t_l that the opinion is expressed would also be apparent in timestamped interactions with the dialog system. Lastly, we choose not to look at the aspect value, a_{ij} in this paper, as we are primarily concerned with sentiment toward entities (rather than aspects of those entities). Aspects could prove useful in the future as the model is used for a broader task: for instance, one may want to extract a students sentiment toward an instructors teaching style or clarity, however these explorations will require a new future data collection with explicit aspect annotations.

The most similar work to ours comes from Mitchell et al. [6]. Unlike this earlier work, we do not use an artificially balanced data set, instead we collect all the utterances from students about whichever entities they choose to talk about. Also, although we limit the entity types to courses and instructors, we do not limit the entity set. Our method is also different from Mitchell et al. [6] in that we do not evaluate subjectivity. All the entities are assigned a positive, negative, or neutral sentiment, there are no entities without sentiment. We make this choice because we are less interested in the neutral or non-subjective class; rather, what is of interest in a dialogue on course recommendations are the things that the student likes or dislikes.

The next most closely related work to this are the tasks of sentiment slot filling and aspect-based sentiment analysis. Slot filling is the task of discovering information about a named entity and storing it in a knowledge source [11]. The 2013 Text Analysis Conference (TAC) had two similar tasks, which were slot filling and temporal slot filling [10]. For the slot filling task systems had to determine the correct value for a set of slots for people and organizations. People contained slots such as “date of birth”, “age”, or “spouse”, while organizations contained slots such as “website”, “founded by”, and “country of headquarters”. For the task of temporal slot filling a system must determine two time ranges. The first time range is a range in which the expressed slot-value was known to begin being a true statement. The second time range is the range in which the expressed fact is known to have ceased being true. The 2014 task is similar to the 2013 slot filling task except for modifications to evaluation rules and the inclusion of geopolitical entities. In the TAC proceedings sentiment slot filling is the task of taking a query opinion holder and query orientation and returning the set of entities that satisfies this condition. In terms of the quintuple we use to represent sentiment, these are related tasks because although slot filling and temporal slot filling are not exactly sentiment tasks, they are concerned with the entity, aspect, and time values. The sentiment slot filling task is concerned with a different subset of this tuple, which are the entity, orientation, and holder.

Aspect-based sentiment analysis has been the focus of recent SemEval tasks as well as a task of TAC [2] [9] [8]. SemEval is a series of evaluations held to further research on tasks in evaluating the semantic meaning of language. The 2014 sentiment task was continued in 2015, and is being continued again in 2016. Researchers submitted a variety of models to evaluate the sentiment of aspects on sets of reviews for laptops, restaurants, and hotels. The highest performing entry in the SemEval 2015 Task 12 aspect polarity classification was a maximum entropy model [5] using bag of words (BoW), lemmas of verbs and adjectives, bigrams that occur after verbs, negation terms, punctuation, the location of polarized terms with respect to punctuation, and features to represent the type of entity being evaluated. The second highest scoring system used a support vector machine (SVM) classifier with features derived from n-grams, pointwise mutual information scores, part of speech tags, parse trees, lexicons, and negation words [13]. The results presented are marked as either constrained or unconstrained systems. Unconstrained systems are allowed to use data outside the training data provided, while constrained systems could not. The top two scoring models were unconstrained but the top scoring constrained system used a logistic regression model. It used lexicon and n-gram based features as in the other models but with the additional Brown cluster features. These are counts of how many words in the sentence belong to each of 1,000 semantic clusters of words derived in previous work [4]. Other entries used similar features with several entries using SVM models and a single entry which chose an unsupervised model.

3 Data

Generally speaking, the most accurate NLP systems rely on the use of machine learning based on data modelling the task at hand. Sentiment analysis is no exception, and previous work has indeed found that data-driven sentiment analysis can perform better than knowledge-based systems [7].

As we are not aware of any dataset consisting of statements describing courses and instructors and the sentiment that the writers (students) have toward them, we gathered our own dataset from a Facebook group where students describe their experiences with classes at the University of Michigan, as well as several surveys given mostly to undergraduate students, with the survey results making up the majority of the data set. We believe this is a suitable data set for the planned dialog system for course recommendations, since the majority of data collected was gathered by asking students to express their thoughts on courses and instructors, in a way similar to how they would express their opinions during a dialog.

The final data set consists of 1,042 utterances written by both graduates and undergraduates, describing both classes and instructors that the students had/interacted with. In total, the utterances include 976 class mentions and 256 instructor mentions, for a total of 1,232 entities. Table 1 shows three anonymized example statements drawn from our dataset.

Student utterance	Annotation
I thought that introductory programming concepts was a difficult class and I did not like it.	⟨class name=introductory programming concepts sentiment=negative⟩
Professor Smith is my favorite teacher that I've had so far.	⟨instructor name=Smith sentiment=positive⟩
I took EECS 203 last Winter. Smith was teaching and I thought the class was excellent.	⟨class id=203 department=EECS sentiment=positive⟩ ⟨instructor name=Smith sentiment=neutral⟩

Table 1. Sample student utterances from our dataset along with annotations. Names of instructors are anonymized.

All the utterances are manually annotated to identify classes, instructors, and the sentiment that the student has toward them. Classes can be mentioned by department and class ID as in "EECS 999," by ID alone as in "999," or by name as in "introduction to artificial intelligence" or "intro to AI." Instructors are mentioned by name, but could be mentioned by first, last, or first and last names.

We model the task of entity extraction using an IOB model, in which each token in the input text is labeled with one of the following three labels: I(nside), O(utside), B(eginning). These labels refer to the position of the token inside an entity. For instance, given the text "I am enrolled in EECS 999.", and assuming the entity to be extracted is a course ID, we would assign the following labels "I$_O$ am$_O$ enrolled$_O$ in$_O$ EECS$_B$ 999$_I$.", indicating that EECS is at the beginning of the course name, 999 is inside a course name, and all the other tokens are outside the course name.

During the annotation, the course ID, course department, and/or course name, as well as the instructor name are extracted and listed explicitly in an XML-style annotation. The perceived sentiment toward an entity is also labeled as either positive, negative, or neutral. When no explicit sentiment is expressed toward an entity, it is assumed to be neutral. If no sentiment is evident from a given utterance, it is assumed to be neutral. Table 1 shows sample annotations for three utterances in our dataset.

To calculate inter-annotator agreement for entity extraction, two human judges labeled 100 utterances from the data set, containing 1,263 tokens. Of these, 1,067 were mutually labeled O. Of the remaining tokens 2% caused disagreement. Including all tokens, agreement is measured as 0.987 using Cohen's kappa. There were two instances where the human judges disagreed on whether or not a sequence of tokens was a name or a description of the course. For example, in the sentence "I believe that databases are a crucial part of computer science and 999 was interesting", while "databases" is part of the class name, one annotator decided that the word was simply a description of the content of the course and not a name.

To calculate inter-annotator agreement for sentiment annotations, two human judges individually labeled all 1,232 entities. The agreement between their annotations was measured at 77.7%, which gives a Cohen's kappa of 0.661 and is considered to be good agreement. Agreement is calculated as the percentage of entities for which both annotators assigned the same label. Of the annotator disagreements, 10.7% were neutral-negative disagreements, 11.2% neutral-positive disagreements, and 0.2% positive-negative.

4 Entity Extraction

We extract entities by labeling each token using the IOB annotation scheme. Three types of B tokens are used; for class ID, class name, and instructor name. However, only two I tokens were used because class IDs are always one token.

For each token in the input text, we build a feature vector, using the following features:

Core features. The initial features we used are simple lexical features not specifically inspired by the task at hand. This provides a baseline with which to compare how effective other features are when added to the model. Table 4 provides the F1 measures for the token set for the features that we experimented with. Features for the current word include the actual word in lowercase, all uppercase, begins with a capital letter, the length of the word, the part of speech category for the word, the part of speech of the word, and the previous two words. Features are also derived from the two words neighboring the current word which are computed the same way they are for the current word.

Lexicons. Lexicon features indicate the presence/absence of a given word in predefined lists of words (lexicons). For this we use two lists. The first list is a list of professor names. This list was manually created from an online faculty listing. The second list is a set of EECS class words. This list of words is created by taking individual words from class names. The two lexicon features are generated for the current word as well as each neighboring word.

Professor titles. We use a list of titles, such as "Dr." or "Prof." to assist with the identification of professor names. The list was compiled manually, and consists of 15 tokens. A feature is generated to indicate whether a token belongs to this list or not. At least one of these tokens occur in 38% of utterances in the corpus.

Sequence and Acronym. Students often abbreviate the names of classes or use a subset of the words in a class name to refer to it. Moreover, there is a high chance that a word that exists in the name of a class or instructor will be followed by another word that also belongs to that name. The sequence feature is a true or false feature that indicates whether the current token comes next in a course or instructor name. Table 2 shows an example sentence and for each token the truth value of the sequence feature.

In	my	first	year	I	took	introductory	programming	concepts	with	John	Smith
F	F	F	F	F	F	F	T	T	F	F	T

Table 2. Example sentence showing when the sequence feature has a true or false value. The list of classes contains “introductory programming concepts” and the list of instructors contains “John Smith”.

The acronym feature is another true or false value that checks to see if there is a class or instructor whose abbreviated name, using at least one of the words, matches the input token. The sequence feature checks to see if the first N words of the current word exist as the beginning of a word that comes after a word in a lexicon that matches on the first M letters of the previous token in the input, where M and N are greater than one. This is a useful feature because we find that students will use phrases such as “intro to artificial intelligence” where the word “introduction” is not fully spelled out. It is also useful to know that a given word exists in order in a class or instructor name. The sequence and acronym features use only the current token because they involve sequences of words we would not gain any extra information from checking the feature for neighboring tokens at the same time.

Nearest Entity. Sometimes class or instructor names are misspelled, and for such cases lexicon features may not be as effective. We create a feature that checks to see if there is a word in the class or instructor name lexicons, excluding stopwords, which has an edit distance less than three to the current token. If a match is found, the feature is set to a value of “C” or “I” respectively. If no token exists, the feature is set to “N”.

For learning, we use a conditional random field, as it has been previously shown to be highly effective for such entity extraction tasks [12]. We run a set of 67-33 train-test splits using stratified sampling across instructors, classes, the five entity tokens, and the three sentiment classes. This ensures that data is well represented for both the extraction and classification tasks and that the same splits can be used for both tasks individually, as well as combined, using the extracted entities as shown in Section 5.

Table 3 shows the results obtained by our system, which makes use of all the features described above. For each of the six token types, we show the precision, recall, and F-score obtained with the system.

	B_{CID}	B_{CN}	I_{CN}	B_I	I_I
Precision	0.978	0.846	0.881	0.915	0.948
Recall	0.987	0.823	0.882	0.929	0.866
F-Score	0.982	0.833	0.881	0.922	0.901

Table 3. Precision, recall, and F-scores for our system for the identification of IOB tokens, for class ID (CID), class name (CN), instructor name (I).

For comparison, Table 4 also shows the results obtained with a basic setting, when only the core features are used; as well as the results obtained with a state-of-the-art entity extraction system available from the Stanford NLP group [3]. Our system shows significant improvement over the individual sequence feature for the I_{CN} token ($p < 0.05$) and for both B_I and I_I ($p < 0.01$) using a Bonferroni correction for multiple comparisons.[1]

System	B_{CID}	B_{CN}	I_{CN}	B_I	I_I
Our system	0.982	0.833	0.881	0.922	0.901
Core	0.983	0.811	0.849	0.863	0.841
Stanford	0.914	0.832	0.840	0.887	0.899

Table 4. F-score figures for the identification of IOB tokens, for class ID (CID), class name (CN), instructor name (I).

To gain a better understanding of the role played by each of the features considered, we also perform feature ablation, where we evaluate the performance of each individual feature when added to the default system consisting of core features. Table 5 shows the results of these experiments. Interestingly, while lexicon features show the greatest improvement, the titles feature does not show any improvement over the core system. It is possible that the relatively small number of tokens in this list provides the model with enough examples of each that from the unigram weights it learns the importance of these words without the need for this additional feature. The sequence, acronym, and nearest entity features are all based on the provided lexicons so it is not surprising that sequence and nearest entity features work well. One of the main drawbacks of using the Stanford system is that the words that exist in class names, which are commonly used words, are not recognized as parts of named entities. This is likely why the sequence feature gives a statistically significant improvement over the core model. The acronym feature appears to be less useful simply because many class names are not commonly abbreviated. The most frequently abbreviated name is “AI” for “artificial intelligence.” Classes are more often referred to by a subset of the words in the class name, which is a case covered by the sequence feature.

Features	B_{CID}	B_{CN}	I_{CN}	B_I	I_I
Lexicons	0.982	0.831*	0.875*	0.915*	0.896*
Titles	0.983	0.811	0.851	0.861	0.839
Sequence	0.983	0.829*	0.871*	0.858	0.832
Acronym	0.983	0.811	0.848	0.860	0.835
Near.Ent.	0.983	0.826*	0.865*	0.910*	0.895*

Table 5. Feature ablation for the identification of IOB tokens, for class ID (CID), class name (CN), instructor name (I). * indicates a feature whose difference from the default feature set is statistically significant with $p < 0.01$ using a paired t-test

[1] Throughout this paper, we measure the statistical significance of our results by using a paired t-test using the same 100 train-test splits

5 Sentiment Analysis

Sentiment analysis is performed as a classification task using three classes, positive, negative, and neutral. For each utterance, each entity extracted by the conditional random field has its sentiment classified. We perform an analysis under the assumption that all entities are perfectly extracted in addition to an evaluation that uses the entities recognized by our system.

Training and test splits are generated one hundred times for each feature, tested individually. Training and test splits contain approximately 67% and 33% of the data respectively. Data was stratified as mentioned in Section 4, by each token type, as well as by entity type, instructor or class, and sentiment class label. An SVM classifier is trained and a grid search for the SVM cost and gamma parameters is performed using three-fold cross validation on the training set.

The default model is constructed using unigram counts. The first step is to extract a set of the words that exist in the training set. Using this vocabulary set, counts are constructed from a given utterance. The counts are then weighted based on their distance, in number of tokens, to the entity in the statement. For each occurrence of each word the feature is computed by $\sum_{i \in I} 1/d_{ie}$, where I is the set of occurrences of that word and d is the distance to the entity, e, being classified. This linear weighting scheme gives the default classifier shown in Table 6. Given this model new features can be developed and added to the model to attempt to improve accuracy.

A sentence is not linear in nature. A sentence contains clauses and phrases which can be grouped into a tree structure. Consider the sentence “I thought the class was going to be good, but it was terrible”. In this sentence “class” is the entity and we find that a positive sentiment word “good” is closer (using linear distance in number of tokens) to the entity than the negative sentiment word “terrible” which represents the actual sentiment toward the entity. If we construct a constituency parse tree from this sentence and calculate distance as the number of hops between nodes in the tree, then the negative sentiment word is actually closer to the entity word. This weighting scheme seemed more realistic but unfortunately gave a lower accuracy as shown in Table 6.

The last feature implemented uses two lexicons, Bing Liu’s sentiment lexicon, and the MPQA sentiment lexicon. These are two of the most commonly used lexicons in recent sentiment work. These lexicons contain lists of positive, negative, and neutral words. The features that represented individual words are now taken as quadruples for each of the sentiment values. Each word’s weight is added to the original group as well as one of the three sentiment groups if it is listed as having one of the three sentiment classes in the lexicon set. If the word does not exist in a lexicon it is only added to the original group. The weight for the word, calculated the same as previously described, is then simply added to its corresponding points in the quadruple. This addition to the feature set gives a significant improvement over the baseline model. This model is shown as the weighted sentiment lexicon BoW model in Table 6.

As a baseline, the Stanford sentiment analysis method can be applied to our data set. This is a method that was designed for sentence level sentiment analysis and assigns an integer score of 0-4 to each sentence. These correspond to “very negative”, “negative”, “neutral”, “positive”, and “very positive” in increasing order. The five values can be mapped to the three values used in our data set in a number of ways, but the way that maximizes the accuracy over our entire data set maps "very negative" to our "negative", "negative" and "neutral" to our "neutral" and "positive" and "very positive" to our "positive". Using the sentence level labels and assigning them to each entity contained within that sentence gives the accuracy shown in Table 6.

Feature	μ Accuracy	σ^2
Default Linear Weighted BoW	67.9%	1.5%
Tree Weighted N-grams	65.6%*	1.9%
Weighted Lexicon BoW	69.5%*	1.4%
Best Guess Baseline	52.8%	-
Stanford Sentiment	62.3%	-
Annotator Agreement	77.7%	-

Table 6. Sentiment accuracies where * indicates a feature whose difference from the default linear weighted BoW is statistically significant with $p < 0.01$ using a paired t-test.

The sentiment analysis classifier can also be evaluated using the output entity tokens from the entity extraction method. This is how the model will be used in practice and it requires a slightly different evaluation. We use the same precision, recall, and F-score measures as previously used but the true positive is defined as the correctly recognized tokens representing an entity and a correct sentiment label. The results of this analysis are shown in Table 7.

Entities	Precision	Recall	F-score
Ground Truth Instructors	0.643	0.643	0.643
Ground Truth Classes	0.710	0.710	0.710
Ground Truth Both	0.695	0.695	0.695
Extracted Instructors	0.581	0.578	0.580
Extracted Class	0.571	0.599	0.585
Extracted Both	0.573	0.600	0.586

Table 7. Class and instructor precision, recall, and F-score measures for full sentiment analysis of entities using entity extraction output is compared to the accuracy of using ground truth extracted entities. When using the ground truth labels, precision, recall, and F-score are the same.

6 Discussion

This section discusses the analysis of features and some of the reasons why our system fails. We perform these analyses through a manual inspection of the errors. The errors that occurred can be grouped into categories. The first is unresolved pronouns. Sentiment is classified by weighting unigrams by their distance to an identified entity in the utterance. If a pronoun is used in the sentence this sometimes leads to a larger distance between the entity token and the tokens which are indicative of a sentiment value. Example: “John Smith taught the data mining class that I took. He was an amazing teacher and I wish that he would teach machine learning.” In this sentence “John Smith” is classified as having neutral sentiment, rather than positive.

There exist utterances for which sentiment is expressed toward one entity and includes one or more other entities close to the sentiment words which only describe the first entity. Example: “I think that John Smith was an interesting teacher in natural language processing”. In this sentence positive sentiment is incorrectly assigned to “natural language processing.”

Negated phrases are notoriously tricky in sentiment analysis. In this case, while we find that there are very few positive-negative disagreements, there are cases where the system believes that a sentence is negated to neutral. Consider “I’m not going to take a theory course again. 999 was not good”. In this sentence human annotators agree that a negative sentiment is being expressed toward the entity “999”, but one could see how a phrase like “X was not great” does not necessarily imply that there is a negative sentiment toward “X.”

There are many instances where the sentiment assigned to an entity is neutral rather than a positive or negative value. This is not particularly surprising since 52% of instances contain this label the data is biased. Most of the annotation disagreements were also between neutral and either positive or negative. Example: “EECS 999 was a lot of work, but one of my favorite classes.” In this example “EECS 999” is assigned a neutral sentiment.

There are a few cases where the entity that should have been identified is missing. In particular there is a professor is missed multiple times because this is an instance of a professor who has a name which does not exist in the lexicon. Some professors have multiple names by which students refer to them. There are also a few instances where a name of a person who is not an instructor is mentioned and incorrectly annotated as an entity.

There are a few errors occur more frequently where annotators disagree. The first is related to annotation disagreements. There is considerable overlap in the errors made by the system with the set of entity annotations for which the two human judges did not agree. One example of this is “I like Smiths’s 999”. Annotators disagreed as to whether the sentiment was directed toward the class or if the sentiment was directed toward the instructor because it is specifically “Smith’s” version that is being mentioned.

There are certain comparative sentences that could possibly be resolved in a larger context, but cannot be resolved as individual sentences. These represent a rather small portion of the data set. An example sentence is “I think I would have enjoyed taking intro to AI with Smith more than when I took it with Jones.” It is not clear that enjoying a class with “Smith” more than “Jones” implies a positive sentiment toward “Smith.” These statements could be resolved over the course of multiple turns in a dialog.

While some negation can be learned from the model by learning that a negation word often co-occurs with a sentiment word it may be useful to develop additional features for handling negation. Coreference resolution is another difficult problem in natural language processing whose application to sentiment analysis could improve the performance of this model. This would be done by identifying pronouns in utterances and determining which mentioned entity the pronoun refers to and factoring this into the weighting scheme for unigram features. The incorrect attribution error could possibly be fixed with a different distance measure. We attempted to resolve this by using the constituency tree weighted grammar. If you generate a constituency tree for the example above you will find that "interesting" is still closer in distance to the class entity than the instructor entity. A dependency parse tree or other graph based construct could reduce these errors. It may be useful to solve an opinion expression problem to try and limit the scope of which words are used to generate features, though this may harm the classifier as it will sometimes exclude crucial information for the evaluation of sentiment.

The entity extraction features and the lexicon features from the sentiment classification were the most useful for both tasks, which is in line with previous work that found domain-specific features to often be the most useful.

In Section 4 we explained that the title feature likely did not provide significant improvement because even though a large number of utterances contained tokens from the title lexicon, the classifier can more easily learn the significance of these words because there are few tokens in the lexicon. The acronym feature is not as useful because few class names are abbreviated. We may find that when gathering more data from undergraduate students there are more abbreviated terms, which would make this feature more useful. The nearest entity feature is useful in combination with the lexicon features because students often make spelling errors, especially with instructor names. The sequence feature allows the classifier to make use of the fact that a lexicon is not a bag of words. It thus helps most with I tokens and more with classes than instructors, as classes can have many words, while instructors usually have 2-3 token names.

The choice to weight words in the sentiment classifier appears to be a good one. Classifiers in most previous work either extract opinion expressions and treat them as a bag of words, limit the relevant words to a window around the aspect/entity, or use hand crafted features based on the order or distance of a subset of the tokens. This weighting approach, similar to Boiy and Moens [1], allows the classifier to learn from all words in the utterance and may capture more subtle or complex expressions.

7 Conclusions

This paper has presented work on targeted sentiment, as an enabler step for natural language understanding in a dialog system for academic advising. More specifically, we addressed the problem of identifying the sentiment held by students toward courses and instructors. The paper explored new sets of features for the extraction of entities and for the classification of sentiment toward these entities. The entity extraction and sentiment classification models showed a significant improvement over the baseline models. The features that directly used domain-specific lexicons were found to have the most significant impact on performance. After discussing the models, an analysis of errors currently affecting the system was presented, laying the ground work for more advanced sentiment analysis in the future.

Future work includes the application of these methods toward other entities mentioned in advising sessions or aspects of those entities. In addition, the steps to address the errors described in Section 6, including coreference resolution, different weighting schemes, and opinion expression identification could be interesting directions to pursue.

Acknowledgments

This material is based in part upon work supported by IBM under contract 4915012629. Any opinions, findings, conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of IBM.

References

1. Boiy, E., Moens, M.F.: A machine learning approach to sentiment analysis in multilingual web texts. Information retrieval 12(5), 526–558 (2009)
2. Ellis, J., Getman, J., Strassel, S.: Overview of linguistic resource for the tac kbp 2014 evaluations: Planning, execution, and results. In: Proc. Text Analysis Conference (TAC2014) (2014)
3. Finkel, J.R., Grenager, T., Manning, C.: Incorporating non-local information into information extraction systems by gibbs sampling. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics. pp. 363–370. Association for Computational Linguistics (2005)
4. Hamdan, H., Bellot, P., Bechet, F.: Lsislif: Crf and logistic regression for opinion target extraction and sentiment polarity analysis (2015)
5. Li, Y., Zhang, Y., Doyu Li, X.T., Wang, J., Zuo, N., Wang, Y., Xu, W., Chen, G., Guo, J.: Pris at knowledge base population 2013. In: Proc. TAC 2013 Workshop (2013)
6. Mitchell, M., Aguilar, J., Wilson, T., Van Durme, B.: Open domain targeted sentiment (2013)
7. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Foundations and trends in information retrieval 2(1-2), 1–135 (2008)
8. Pontiki, M., Galanis, D., Papageogiou, H., Manandhar, S., Androutsopoulos, I.: Semeval-2015 task 12: Aspect based sentiment analysis. In: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, Colorado (2015)
9. Pontiki, M., Papageorgiou, H., Galanis, D., Androutsopoulos, I., Pavlopoulos, J., Manandhar, S.: Semeval-2014 task 4: Aspect based sentiment analysis. In: Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014). pp. 27–35 (2014)
10. Surdeanu, M.: Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In: Proceedings of the Sixth Text Analysis Conference (TAC 2013) (2013)
11. Surdeanu, M., Ji, H.: Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In: Proc. Text Analysis Conference (TAC2014) (2014)
12. Zhang, L., Liu, B.: Aspect and entity extraction for opinion mining. In: Data mining and knowledge discovery for big data, pp. 1–40. Springer (2014)
13. Zhang, Z., Lan, M.: Ecnu: Extracting effective features from multiple sequential sentences for target-dependent sentiment analysis in reviews (2015)