A low-resources approach to Opinion Analysis: Machine Learning and Simple Approaches

David Kirk Evans
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
devans@nii.ac.jp

Abstract

In this paper I present a system for automatic opinion analysis built in a short time-frame using freely available open-source processing tools and lexical resources available from prior research. I use a simple feature-set that is largely language independent and a freely available machine-learning framework to model the subtasks as classification problems and report on my system's performance. Additionally, I show that blind relevance feedback improves results sentence-level relevance judgment. My system shows that it is possible to quickly build an opinion analysis system in a short period of time that can perform at an average level.

Keywords: *NTCIR, Opinion Analysis, Machine Learning*

1 Introduction

The NTCIR Pilot Task on Opinion Analysis is a multi-language sentence-level granularity opinion identification task with a opinion holder identification component, and optional polarity and sentence-to-topic relevance components. Please see [1] for more details about the task, including a description of the corpus and overview of the results.

As a co-organizer of the NTCIR Pilot Task on Opinion Analysis, I developed a system to submit to the evaluation. I have not worked previously in the area of Opinion Analysis, but have had extensive experience in summarization and text similarity computation, so decided that I would view this task as an investigation into how far fairly simple techniques combined with off-the-shelf machine learning approaches can get you with a relatively short development time.

Since I am also interested in multilingual aspects of computational linguistics, I felt that using surface-level feature-extraction combined with machine learning would allow me to easily use the same approach for multiple languages. I developed an English system, but plan to adapt the system to Japanese and Chinese trained over the data from this pilot task.

1.1 Related Work

Early work in subjectivity analysis focused mainly on determining whether sentences or documents are subjective or objective. Wiebe and her colleagues have performed much of the early work in this area, in [10] describing a corpus tagged at the sentence level for subjectivity and a Naive Bayes classifier using syntactic classes, punctuation, and sentence position as features. Hatzivassiloglou and McKeown [3] introduced the approach of using bootstrapping to determine the orientation of adjectives, then Hatzivassiloglou and Wiebe [4] showed that gradable adjectives are useful for subjectivity classification, and Riloff and Wiebe [6] identified strong subjectivity clues and weak subjectivity clues and surveyed syntactic patterns to detect subjective sentences. Yi et al. [11] present an approach that extracts sentiment towards a given subject instead of classifying entire documents or sentences as positive or negative.

More recently, Pang and Lee [5] examine human performance over the task of rating sentences on a semantic scale (e.g., from one to five "stars") and then develop a classifier for that task. Yu and Hatzivassiloglou [12] develop a Bayesian classifier for determining subjectivity or objectivity at the document level, and a multi-classifier system that uses uni/bi/tri-grams, part of speech frequency, and orientation adjectives as features for sentence classification.

2 Approach

I decided to use machine learning learning techniques to build my system. The general approach is to model the problem as a standard classification problem. I used the WEKA machine learning toolkit [2] to experiment with a variety of classifiers over the features for each task.

The MPQA corpus[1] is used as training data for each of the learners. Resources used to generate the features include the OpenNLP tools[2], for part-of-speech and named entity tagging, a list of country names, a list of names separated into gender from the US Census[3] and the word polarity list presented in previous work by Yu and Hatzivassiloglou. [12]

2.1 WEKA Machine Learning Environment

Each of the experiments were conducted using the WEKA machine learning environment. The WEKA environment contains approximately 40 machine learning classification algorithms. I approach Opinionated sentence identification, polarity identification, and opinion holder identification as three separate machine learning problems, generate training data from each based on the MPQA corpus, and then train a variety of classifiers from the WEKA environment. For each task I choose one of the top performing classifiers based on the ten-fold cross-validation evaluation results, then integrate the classifier and its learned model into a prediction framework. Details on the features and classifiers used for each of the tasks are given below.

2.2 Opinion Identification

Sixteen features are used for opinionated sentence identification: length, average_valence, max_valence, min_valence, in_quote, has_person, has_org, has_place, has_country, he_count, she_count, he_said, she_said, ne_said, said_ne, and location. While most of the features should be self-explanatory, some might not be as obvious. Table 3 describes the features used for the Opinion Holder subtask, many of which are used in the opinion identification and polarity subtasks as well. Valence is the sum of the polarity scores from the opinionated word list, obtained from the authors of [12]. Their approach basically builds a large set of opinionated words and their polarity (positive or negative) based on bootstrapping from a smaller list of words. The features he_said, she_said, ne_said, and said_ne are the number of times that "he said", "she said", or some named entity was found near a communication verb, and location is the sentence position, where 1 would be the first sentence of the document with larger numbers sentences that occur later in the document. All of these features are fairly simple to compute, with only the named-entity and opinion word valence features requiring language-specific information.

Sentences from the MPQA corpus were used as training, however their annotation format is at a finer granularity and had to be converted into sentence-level "subjective" or "not subjective" tags. This was done according to the rules given in Section 4 of the Database.1.2.README file. These are the same rules used in [7, 6].

Classifier	Accuracy	Precision	FP
Baseline	54.29	54	508
JRip	62.83	65	230
Logistic Model Tree	64.07	65	228
BayesNet	63.55	66	218
SMO	63.18	66	210

Table 1. Accuracy, Precision, and False Positive rate of opinion identification classifiers.

I experimented with both nominal and numeric targets for the learning algorithms. The training data assigned a value of "0.0" to non-opinionated sentences, and "1.0" to opinionated sentences for numeric sentences, or the classes "opinionated" and "not" for opinionated and non-opinionated sentences respectively. I tested a total of 27 classifiers, with results from the baseline (majority class) and four top performing classifiers shown in Table 1.

I selected the SMO classifier, which implements John C. Platt's sequential minimal optimization algorithm for training a support vector classifier using polynomial or RBF kernels. The SMO classifier has comparable performance to the BayesNet classifier, but slightly fewer false positives.

For predicting whether unknown sentences are opinionated or not, a Java program is called that computes features for the sentences, the resulting features are parsed into a WEKA-compatible input file using a perl program and the SMO model is run over the input to generate predicted values for each sentence. When converting the predicted values to the binary Y or N labels, any sentence with a predicted value over the arbitrary threshold of 0.8 was labeled as opinionated. Given more time and training data, I would like to run some experiments to tune this value to try to improve performance of the opinionated prediction component of my system.

2.3 Polarity

I trained 15 classifiers for the polarity classification problem. The training data was filtered to contain only opinionated sentences, with polarity values were mapped to (-1.0, 0.0, 1.0) based on the polarity of the sentence (Negative, Neutral, Positive). Polarity for a MPQA sentence is computed by summing the polarities for the subjective elements annotated for a sentence, with a positive polarity as +1, a negative po-

[1] http://www.cs.pitt.edu/mpqa/

[2] http://opennlp.sourceforge.net/

[3] http://www.census.gov/genealogy/names/names_files.html

larity as -1, and 0 for neutral polarities. A sentence with a sum of 0 is labeled neutral, or positive or negative based on the sign of the polarity sum. To map a predicted polarity back to one of positive, negative, or neutral, the thresholds of 0.5 and greater map to a positive sentence, between -0.5 and 0.5 is a neutral sentence, and less than -0.5 is a negative sentence. These thresholds were arbitrarily set, without any tuning based on the MPQA corpus. There were 6054 training instances overall. The feature set is the same as that used for opinionated sentence prediction.

Of the 15 classifiers tested, the Weka instance-based K-nearest neighbor classifier weka.classifiers.lazy.IBk had the best performance, and was selected as the classifier for the polarity component.

2.4 Opinion Holders

The opinion holder identify task is slightly more difficult because in the English task, an opinion holder can potentially come from anywhere in the document, or even not be mentioned in the document at all. I decided to model the task by enumerating all previously mentioned named entities for a given sentence, and then model the learning problem as a classification problem where features are generated for each named entity, sentence pair. The pairs are classified as one of "not holder", "weak holder", "medium holder", or "strong holder" based on the opinion holder from the MPQA corpus. Twenty-four features are used for a named entity-sentence pair. The features are listed in Table 3. The sentence "gender" is computed by summing the values of the genders of the NEs contained in the sentence, where "he", "his", and known male names contribute a positive score, while "she", "her", and known female names contribute a negative score. The genderMatch feature looks to see whether the candidate NE matches the gender of the sentence for NEs that are known gender names or pronouns.

The training data is created by creating a training instance for each sentence and all previous potential opinion holders, and matching the candidate opinion holders to the opinion holders labeled for the MPQA data. A score for each opinion holder matching a holder in the MPQA data is set for the sentence-opinion holder pair as $\frac{1}{\#OpinionHolders}$. This resulted in 180,777 training instances, sampled down to 56,144 instances. In the classifiers trained with that data set, the location (sentence position) feature correlated too strongly, resulting in only opinion holders from the first sentence being extracted. That feature was removed, and new models were trained over 64,054 instances.

I selected the weka.classifiers.functions.Logistic classifier for this problem. Since there is a large number of potential opinion holders for each sentence, I

Feature	Description
location	Sentence position
closeness	Distance from sentence to NE
isPronoun	1 if the NE is a pronoun, 0 otherwise
genderMatch	1 if the "gender" of the sentence matches the gender of the NE, 0 otherwise
numAppearances	Number of times the NE has appeared in the document
prevAppearances	Previous number of times the NE has appeared in the document
isPerson	1 if the NE is a person
isLocation	1 if the NE is a location
isCountry	1 if the NE is a country
isOrganization	1 if the NE is an organization
inQuotes	1 if the NE appears inside quotes
isImplicit	1 if the NE is the implicit author NE
neSaid	1 if the NE appears before the verb "to say"
sentNumPronouns	number of pronouns in the sentence
sentNumPeople	number of people in the sentence
sentNumLocations	number of locations in the sentence
senNumOrgaizations	number of organizations in the sentence
sentHeSaid	1 if the sentence contains "he said"
sentSheSaid	1 if the sentence contains "she said"
sentHeTokens	number of "he" tokens in the sentence
sentSheTokens	number of "she" tokens in the sentence
sentNESaid	1 if there is a NE "to say" sequence in the sentence
sentSaidNE	1 if there is a "to say" NE sequence in the sentence

Table 3. Features used in the Opinion Holder classifier.

predict as →	strong	medium	weak	not	total
strong	21987	320	247	4850	27404
medium	7959	519	319	3475	12272
weak	4802	282	713	4830	10627
not	3504	80	95	9916	13595

Table 2. Opinion Holder training results.

post-process the predictions to reduce the number of potential opinion holders. Scores are assigned to each potential opinion holder based on the predicted class of the opinion holder (not, weak, medium, or strong) and some heuristics are used to merge potential opinion holders with similar names, adding to the score of the merged candidate opinion holder. Since I did not create a model to predict the number of opinion holders for each sentence, I decided to randomize the number of opinion holders selected for each sentence by selecting 1 + rand(2), or 1-3 opinion holders per sentence. A future improvement would be to model a classifier to predict the number of opinion holders for each sentence, or to just select one opinion holder for each sentence, as a later examination of the gold-standard data showed that the majority of sentences have only one opinion holder.

Table 2 shows the confusion matrix of prediction results over the training data. The classifier does not predict the medium or weak classes with much frequency, instead favoring either the strong class, or not an opinion holder class.

2.5 Relevance

The relevance task was not modeled as a machine learning problem, since no training data was available, outside of the one sample topic distributed to participants. To determine whether sentences are relevant to the topic or not, a standard vector space model with tf*idf weights [9] is used to match sentences to the topic. The standard formulation for cosine similarity, shown in Equation 1 is used.

$$Sim(Q, D_i) = \frac{\sum_j w_{Q,j} w_{D_i,j}}{\sqrt{\sum_j w_{Q,j}^2} \sqrt{\sum_j w_{D_i,j}^2}} \qquad (1)$$

Due to the sparse nature of text short length of sentences as information retrieval “documents”, Rocchio blind relevance feedback [8] is used to improve relevant sentence detection. The Rocchio term updated formula is shown in Equation 2, where $\beta = 0.75$ and $\gamma = 0.25$, both common values used in the IR community.

$$Q_1 = Q_0 + \beta \sum_{k=1}^{n_1} \frac{R_k}{n_1} - \gamma \sum_{k=1}^{n_2} \frac{S_k}{n_2} \qquad (2)$$

Rocchio blind relevance feedback modifies weights for query terms based on the assumption that good matches to the query will contain additional terms that are also indicative of matching content, and sentences that do not match will contain other terms that are indicative of non-matching content. Since sentences are very short units of text to use in vector space models for similarity, I thought it would be important to draw in additional relevant terms over just the ones in the query through the process of blind relevance feedback. In my experiment, I perform one round of feedback for each document by updating query terms from the top three matching sentences plus any sentences with a cosine similarity greater than 0.15, and reducing weights for terms from sentences with a similarity of less than 0.01. After performing the term re-weighting for each document, the actual retrieval is performed, and sentences with a similarity over the threshold 0.15 are considered relevant. Since I did not have any training corpus to tune this parameter, it was set by hand based on runs over the single sample topic provided for English.

3 Results

For the evaluation approach and detailed results, please see [1]. Table 4 shows the results for the lenient and strict standards over the opinion identification, polarity, and relevance subtasks. In addition, **unofficial** results for a run that does not use Rocchio blind relevance feedback are included. Table 5 shows the results for the opinion holder evaluation.

3.1 Opinionated

In the lenient evaluation my system performs very well from the standpoint of precision, performing poorer that only one system, and average from the recall and f-measure standpoints. I’m encouraged by these results as my system still have many areas that can be tuned with further training data, and the mismatch between the MPQA and NTCIR Opinion corpora also makes it more difficult to perform well when

Table 4. English Opinion Analysis results

Standard	Opinionated			Relevance			Polarity		
	P	R	F	P	R	F	P	R	F
Lenient (official)	0.325	0.624	0.427	0.510	0.322	0.395	0.077	0.194	0.110
Strict (official)	0.073	0.642	0.131	0.242	0.355	0.287	0.014	0.185	0.027
Lenient (no Rocchio)	0.325	0.624	0.427	0.490	0.163	0.245	0.077	0.194	0.110
Strict (no Rocchio)	0.073	0.642	0.131	0.236	0.183	0.206	0.014	0.185	0.027

working with data that is different from the training data.

For the strict evaluation, overall all precision scores decrease dramatically, but my system still does quite well from the standpoint of precision and f-measure, performing worse than only one system. It has average recall, performing better than four systems, and worse than four systems.

The strict evaluation is difficult over the English data as there are a very small number of examples for the strict evaluation: the English data does not have high agreement between all three annotators, making it difficult to learn a classifier that can perform well for the strict standard. It would be interesting to increase the number of annotators to some larger number, such as eleven, and define multiple standards based on the number of votes from each annotator.

3.2 Relevance

Only six of the nine submitted runs contained relevance information, and of those six runs two of the groups runs did not differ in their relevance evaluation, giving four unique relevance runs. Under the lenient standard, my system had the best performance for precision, third best performance for recall, and second best performance for f-measure. Under the strict standard my system had the best precision and f-measure, with the third best recall.

As an unofficial experiment, the system was re-run without using blind relevance feedback, shown in Table 4 as the two runs labeled (no Rocchio). In both strict and lenient evaluations, precision and recall fell in the runs that did not use Rocchio blind relevance feedback, confirming our hypothesis that blind relevance feedback helps improve relevance results for sentence-level bag-of-words cosine similarity metrics.

3.3 Polarity

Under the lenient evaluation, my system performed poorly overall, better than only one system in each category. Under the strict evaluation relative performance does not change.

3.4 Opinion Holders

Table 5 shows the results for my system on the opinion holder identification subtask.

Table 5. English Opinion Holders Analysis results

Standard	P	R	F
Lenient	0.066	0.166	0.094
Strict	0.018	0.169	0.032

For the opinion holder identification subtask, my system performs poorly. A key reason for this is that my system submits are large number of opinion holders: from one to three for each sentence. The actual annotated data has, on average, a little over one opinion holder per sentence, so a better strategy is to propose only one opinion holder per sentence. Despite offering multiple opinion holders per sentence, my system's recall is still quite poor.

A second large factor in the poor opinion holder identification is the feature set actually used for training the classifier. Originally the "sentence position" feature was included in the feature set used for training, and an examination of the resulting classifiers showed that the classifiers would **only** select opinion holders from the lead sentence of the article when the feature was available. I intuitively did not think that was a good strategy, so I removed the feature from the training set and re-trained the classifiers. While that hurt performance over the training data, I felt that it was a more general approach to learn rules that would take into account other, more complicated features other than solely the sentence position.

4 Conclusions

I am pleased with the performance of my system, which received average scores despite being this researcher's first foray into the opinion analysis field. The system was developed using a freely available open-source tagging framework, some lexical resources from previous work in the area, and a machine learning framework to create classifiers trained over data from a different corpus.

While my system performed average in most subtasks, it also performed better than average in precision for opinionated sentence detection, but disappointingly performed quite poorly for polarity and opinion holder identification. Due to the different nature of the training and testing corpora, I believe that the machine learning approach would do much better with training data similar to the evaluation data, but it still seems that the MPQA corpus was beneficial in creating classifiers for use on a different corpus. There are also many parameters that can be tuned for my system.

While the overall performance of my cosine-similarity based relevance component was not as good as I had hoped, it is clear that the implementation of blind relevance feedback improved scores for the relevance component, which can be merged with other features beyond a bag-of-words model to determine relevance in the future.

Since the majority of the features used in my system are quite simple, I plan to create a Japanese version of this system in the near future and run an evaluation to see how well the system can be ported to another language.

References

[1] H.-H. Chen, Y. Seki, and D. K. Evans. Overview of opinion analysis pilot task at ntcir-6. In *Proceedings of the 6th NTCIR Workshop Meeting*, Tokyo, Japan, 5 2007.

[2] S. R. Garner. Weka: The waikato environment for knowledge analysis, 1995.

[3] V. Hatzivassiloglou and K. R. McKeown. Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

[4] V. Hatzivassiloglou and J. M. Wiebe. Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the 18th International Conference on Computational Linguistics*, 2000.

[5] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, pages 115–124, 2005.

[6] E. Riloff and J. Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing (EMNLP 2003)*, pages 105–112, Sapporo, Japan, July 2003.

[7] E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL-03)*, 2003.

[8] J. Rocchio. Relevance feedback in information retrieval. In *The Smart Retrieval System — Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[9] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[10] J. Wiebe, R. Bruce, and T. O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th Association of Computational Linguistics*, pages 246–253, 1999.

[11] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack. Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In *The Third IEEE International Conference on Data Mining*, pages 427–343, November 2003.

[12] H. Yu and V. Hatzivassiloglou. Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, volume 10, pages 129–136, Morristown, NJ, USA, 2003. Association for Computational Linguistics.