Estimating Sentiment in Eli

Computational Analysis of Tone in Student Responses to Student Writing

Michael Wojcik
Rhetoric & Writing
Michigan State University
East Lansing, Michigan

wojcikm4@msu.edu

ABSTRACT

Eli is a web-based application for coordinating and assessing writing review, typically used by students in English composition classes at the secondary or undergraduate level. It provides students and instructors with information about the reviews that students write of one another's writing, based on reception (whether the original author finds the review helpful), but does not currently attempt to compute any metrics based on the content of the reviews themselves. I describe a prototype of a proposed addition to Eli which employs sentiment-determination techniques to analyze the rhetorical tone—the attitude of the writer toward the material—of reviews. I describe how this information may be useful for instructors, and how the introduction of NLP techniques into an application like Eli can extend its capabilities; and discuss what sentiment analysis might tell us about writing review.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text analysis—*sentiment determination*

General Terms

Algorithms, Documentation, Human Factors

Keywords

Writing review, sentiment analysis

1. INTRODUCTION

Eli, a web-based application for coordinating and assessing writing review, is an innovative approach to teaching (writing) composition, particularly at the secondary and undergraduate levels, and potentially a useful tool for professional writing management as well. It grew out of work at the Writing in Digital Environments (WIDE) Research Center at Michigan State University which was aimed at applying the tools of "social media" websites to contemporary composition pedagogy, and is now being developed into a commercial product for the education market (WIDE, 2011). Eli provides writers, typically students in writing classes, with an environment in which they can read and review one another's writing, and then provide feedback on the reviews they receive in turn, so they can become better reviewers as well as better writers. Research in language-composition pedagogy has shown that peer review coupled to a process for improving reviewing skills is one of the most effective ways to improve a writer's performance; Eli is designed to streamline this process.

Eli also has potential applications outside the classroom. If we believe review is useful for documents that will be circulated, such as product documentation—and of course such documents are often reviewed by technical experts, usability professionals, team supervisors, etc—then it seems plausible that a system for improving writing review could be effective and economical in the workplace as well. Content-producers would occupy the role currently filled by students; reviewers might be other writers, peers working in other aspects of product development, supervisors, testers, end users, etc. In a professional writing environment, review parameters (the equivalent of review "assignments") might be generated by supervisors, the writing team, usability experts, or the authors themselves. It could also be useful in open-source development projects, where social relations between authors and reviewers are more tenuous (and language fluency may be more varied).

In the Eli environment, students can post their own writing, then read other students' texts and write free-form reviews of them. (They may also respond to more structured review questions set by the instructor, such as Likert-scale ratings and checklists of requirements. Instructors can provide "prompts" to guide students in writing their reviews.) As students receive reviews on their own writing, they can evaluate those reviews, and develop plans for revising their own writing. Meanwhile, Eli shows them how the reviews they created were received by their fellow authors. So each Eli user does four kinds of writing: primary text, reviews, responses to reviews, and revision plans. Part of Eli's function is simply to provide space for and coordinate all of these activities. Beyond that, though, it tracks user actions such as creating and viewing objects (primary texts, reviews, etc), highlighting or copying text, commenting, and so on.

Eli calculates a *helpfulness metric* (now the subject of a patent application by the university) which gauges the utility of a review based on the receiving author's actions and feedback (Hart-Davidson *et al.*, 2009). For example, when an author copies a suggestion from a review onto the revision plan for the next revision of the writing assignment, Eli notes that action and increases the reviewer's helpfulness score. Eli maintains a cumulative helpfulness score for each user (student), based on the average helpfulness of that user's reviews. Instructors use this information to track student performance as a peer reviewer, and students use it to improve their reviewing skills. However, Eli currently does not attempt to compute any metrics based on the actual *content* of reviews, only what the reviewed author does with them.

Here I describe an experiment in applying Natural Language Processing (NLP) techniques to review content in Eli, as the first stage in a project to explore the utility of NLP for calculating additional metrics about reviews and helping instructors. Specifically I was interested in two questions. First, is UIMA, a standard, open-source framework for processing unstructured data such as Eli's review documents, a good fit for the Eli system? Second, can we apply sentiment-analysis algorithms to reviews in Eli and extract useful information? I will also discuss why automated sentiment determination might be useful for Eli and speculate on some other NLP applications for Eli for the future.

2. NATURAL LANGUAGE PROCESSING WITH UIMA

UIMA, the Unstructured Information Management Architecture, is a standard for and implementation of a framework for processing unstructured data. In principle UIMA can be used with text, audio, images, video, and so on, but most UIMA applications to date have operated on text. UIMA applications take many forms and serve many purposes, but broadly speaking they are usually designed to do one thing: add structure to unstructured information by analyzing it and creating metadata based on that analysis. In the realm of NLP, that includes basic structural tasks such as tokenizing, chunking, parsing, but it can also include higher-level abstractions like discerning sentiment. UIMA itself consists of a modular processing architecture (including support for distributed and loosely-coupled flows) and a data schema for representing subjects, analyses, annotations, and similar objects of interest. There are also many extant free, open-source UIMA modules that a UIMA developer can make use of.

UIMA was originally developed by IBM, then taken up by the web-standards organization OASIS for standardization. IBM handed the code base off to the Apache Foundation for maintenance; while anyone can implement UIMA, in practice Apache UIMA is the implementation most people use. Apache UIMA is available in source and compiled form and comes with a number of useful sample modules, plus documentation, tutorials, and the like. It is designed primarily for use with C++ and Java, and includes enhancements to the Eclipse IDE for those languages, but it supports other programming languages through bridge interfaces. Programs written using Apache UIMA can be packaged as modules for use in larger projects, as standalone applications, as web services, or as worker roles for distributed systems such as Hadoop or Apache's own UIMA-AS (UIMA Asynchronous Scaleout).[4]

UIMA is useful for analytic NLP tasks in a number of ways. It's easy to represent common NLP functions such as part-of-speech tagging as UIMA annotators, which can attach arbitrary metadata to spans of input data in a fashion that's easy for downstream modules to consume. The modular architecture of UIMA makes it easy to build processing pipelines of these basic NLP functions, freeing experimenters up to focus on problems of specific interest. Because UIMA is standardized, popular, and free, it's convenient for sharing work with other researchers; its support for loose coupling (e.g. deploying applications as web services) makes it easy to provide other researchers with a sandbox in which to try out a new feature. Its scale-out capabilities help with processing large corpora of data, and with very compute-intensive research problems. And many NLP modules are available for UIMA: the Apache distribution includes a tokenizer and sentence delimiter, for example, and there are wrappers for the basic functions of the OpenNLP suite.

For this experiment, I created a number of UIMA modules and configurations (what UIMA calls "descriptors") to experiment with sentiment analysis of review data extracted from Eli. I ran these under the UIMA Document Analyzer application, a convenient utility for UIMA experimentation. In the longer term, if we find this kind of work is useful for Eli, we will more likely expose it as a web service and integrate it loosely into the Eli system. That will permit independent development on Eli and these NLP features for it. We would likely also support both online and batch-mode execution, so that Eli users could instantly get results for a select set of reviews, or process a larger corpus of them in the background.

3. SENTIMENT ANALYSIS

Automated sentiment analysis is the process of detecting the expression of opinions, feelings, and evaluations in text, and determining something about what position they express—typically simply if they express a *positive* or *negative* sentiment. Sentiment analysis tasks may also include identifying the topic of discussion (what the sentiment applies to), and distinguishing "authentic" sentiment-bearing texts from "fraudulent" ones (such as product reviews written for hire on sites like Amazon.com).

The literature on sentiment analysis (including sentiment classification, extraction, retrieval, aspect identification, etc) is large. This is a popular area of NLP research, no doubt in part because of its commercial value: it is used by media-analysis professionals in government, public relations, and similar fields; by market analysts studying online product reviews; and in other well-funded areas.

As with most areas of NLP, there are numerous approaches to sentiment analysis. Some simply look for sentiment-bearing words (a *bag-of-words* language model); others try to take linguistic structure into account. (A well-known problem with the bag-of-words approach is detecting *polarity-reversal*, when the sentiment of a term is reversed by the phrase it appears in: "I don't dislike. . . ".) Some incorporate the results

of linguistic analysis, for example dictionaries with category labels assigned to words by linguists, while others attempt to derive lists of sentiment-bearing terms by analyzing a corpus of documents. These latter *machine-learning* algorithms may be *supervised*, indicating they use metadata assigned by human readers to recognize sentiment in the training data; or *unsupervised*, using some other mechanism (e.g. a numerical "star" rating accompanying a textual review) as a heuristic to gauge sentiment; or *minimally-supervised*, starting with a small set of "bootstrap" information about sentiment and expanding it by finding correlated information in the corpus.

While simple bag-of-words sentiment analyzers can produce interesting, if not terribly accurate, results in many domains, current research on sentiment analysis includes a wide range of very sophisticated approaches, often combining multiple methods. For example, Nakagawa *et al.* [11] describe a system that uses one linguist-created dictionary to identify sentiment-bearing terms, a second dictionary derived from a different linguistic project to identify polarity-reversing terms, a parser to split sentences into phrases, an algorithm to create a "dependency tree" of those phrases to determine when a polarity-reversing term applies to a given sentiment, and a complex machine-learning algorithm to assign values to hidden variables that represent combinations of numerous features (polarity of a given phrase, part-of-speech of a sentiment-bearing word, etc) to construct a model that maximizes the probability of determining the correct sentiment. See [20] for a longer discussion of the sentiment-analysis algorithms I reviewed when starting this project.

3.1 Sentiment and Rhetorical Tone

When this project was first broadly conceived sometime in 2008, it was originally defined in terms of *tone* rather than sentiment (no doubt because it was conceived by rhetoricians rather than linguists). Rhetorical tone is a more abstract and complex concept than simply sentiment. Tone is traditionally defined as something like "the perceived attitude of the author toward the subject", and "attitude" clearly encompasses dimensions that exceed at least the reductive positive/negative sentiment model. An attitude can be positive and also enthusiastic, paternalistic, envious, cautious, and so on, for example. And a broad definition of sentiment would include aspects of a text that are not part of tone. Part of my larger project, which falls into the area I call computational rhetoric [19], is to investigate to what extent we can heuristically determine rhetorical tone based on extracted features such as sentiment.

At this stage of the project, however, I am using sentiment as a proxy for tone. If I find we can estimate simple sentiment at a useful degree of accuracy in the Eli review data, then we can move forward with more complex sentiment estimation and modeling rhetorical tone.

I believe this information has at least three possible uses. First, instructors may find that students respond differently toward reviews based on their tone, and use that information to make suggestions to students about how to couch their review comments. Second, review-tone information might be used to flag certain problematic reviews for instructor attention. Third, tone analysis of reviews could begin to address open research questions such as whether the helpfulness of a review is correlated to its tone or perceived sentiment.

3.2 Sentiment Analysis of Writing Reviews in Eli

Because student-writing reviews are a more focused genre than many of the domains where sentiment determination is currently popular (for example, product comments forums for online shopping sites), and since the writers feel some constraint to produce well-formed text, the input has relatively low noise. In fact, I initially expected to have to implement heuristic spelling correction (using a lexicon of English words and a Minimum Edit Distance algorithm) to avoid confusing the sentiment-detection algorithms with misspellings and typographical errors, but after reviewing the data extracted from Eli I determined that the error rate was quite low. I believe that this is at least partly due to the reviewers' sense that they are writing in a writing-class environment, encouraging them to take extra care; but I have not tested that hypothesis, and there are certainly other possible explanations, such as the spell-checking feature built into many web browsers. (Eli is a web application, so the user interface is implemented in the browser.)

On the other hand, sentiment markers in student-writing reviews also tend to be more subtle. Students are generally reluctant to express overtly negative opinions of one another's writing, particularly when author and reviewer are part of a small group such as a class. They may try to couch their negative opinions "constructively" or in an ostensibly positive gloss, so e.g. "your second paragraph doesn't make sense" might become "I think your second paragraph might be better if you explained your ideas more". The reviews I studied show a marked tendency to prefer evaluations in the form of factual statements ("your paper was short") over emotional responses which would more directly express sentiment, and to a lesser extent narrations over directives ("I couldn't find your introduction" rather than "you need to write an introduction"). They also often preface their substantive remarks with formal expressions of approval ("overall I liked your essay"), as a sort of social nicety; this will distort a straightforward sentiment evaluation. Again, though, these tendencies were not as marked among reviews I studied as I had initially feared, and a good fraction of them were quite blunt.

One advantage of the Eli data for sentiment analysis is that the process and software guarantee that we always know the subject of a review—that is, which original piece of writing it describes. Also, reviewers almost always confine their remarks to the writing they're reviewing, without referring to other texts (as is typical in, for example, book reviews) for purposes of comparison or illustration. This relieves us of one of the tasks often faced by sentiment analysis, determining the subject of sentiment.

4. APPROACHES

The data for the study was extracted from a snapshot of the Eli database that was generated during beta testing. For ease of processing, I unloaded the relevant columns from the database to text files, rather than operating on data in the database. Each file contained the database unique row

ID for a review (for identification purposes; this ID was also used as the file name), the numerical rating assigned to the original text by the reviewer, and the text of the review. The extraction process skipped rows that had no review text and some test rows that contained dummy reviews (*lorem ipsum* text). It also removed HTML tags from the review text and converted some HTML entities and non-ASCII characters to their ASCII equivalents. This preprocessing was done with an *ad hoc* program written in GNU Awk.

After extraction and filtering, there were 3659 review documents. They have a wide range of length (from a few words to multiple paragraphs), writing style, and content. Some are simply factual information ("could not open document") or short subjective evaluations ("too technical?"). Others, however, express relatively complex critiques with substantial sentiment markers. There is some use of slang and abbreviations, and the occasional misspelling, but in the main the writing is relatively standard and amenable to processing.

The numerical rating that accompanies each review is an overall quantitative evaluation of the piece being reviewed, in the "four of five stars" mode. In theory, numerical ratings range from 1 (lowest) to 5, but in fact none of the reviewers in this database snapshot had ever assigned a "5" rating. There are likely various reasons for this; for example, students participating in the review process feel charged to provide helpful commentary, which implies that the piece they're reading has room for improvement, and so can't be given the highest possible rating.[1] These numerical ratings are of interest for various reasons. On one hand, we would like to discover whether there's any correlation between numerical rating and computed sentiment—does negative sentiment always mean a low rating? On another, if we assume there is such a correlation, we can use ratings to assign reviews to negative-sentiment and positive-sentiment buckets for training purposes. In fact, this technique was used, as described below.

After extraction, the set of documents was partitioned in various ways for further analysis. First the documents were grouped according to numerical rating. Then 100 each of rating-1 and rating-4 documents were set aside for training purposes. A subset of 10 documents (5 rating-1 and 5 rating-4) were copied from the training set to test how well the sentiment analyzers performed against data they had been trained on. I evaluated sentiment manually in these 10 so that we were not relying simply on the numerical ratings (which did not contain sufficient information anyway, as discussed below). I also took another 40 documents, 10 of each rating, and manually evaluated sentiment in them; these 40 plus the previous 10 became the gold-standard set used to evaluate analyzer performance.

4.1 Representing Sentiment

Early sentiment analysis work often simply represented sentiment as either "positive" or "negative". Some later approaches, such as that of Gamon and Aue [7], added a "neutral" category for documents that don't appear to include any significant sentiment. Other projects have used more sophisticated representations for sentiment, including the "strongly" and "weakly" positive and negative tags in the MPQA corpus [17].

[1]I'd like to thank Doug Walls, one of the original Eli researchers, for a useful discussion of this phenomenon.

positive sentiment?	negative sentiment?	value	text label
no	no	0	neutral
no	yes	1	negative
yes	no	2	positive
yes	yes	3	mixed

Table 1:

For this project, I defined positive and negative sentiment as independent binary features: a document could have no significant sentiment (aka "neutral"), primarily positive sentiment, primarily negative sentiment, or both positive and negative sentiment ("mixed"). It's also possible to express this as a single two-bit integer; I assigned the positive sentiment feature to the more-significant bit, giving four values (1).

For purposes of computing precision and recall when evaluating sentiment-analysis implementations, I treat neutral as the lack of any features, and calculate separate results for positive-sentiment alone, negative-sentiment alone, and the two combined.

4.2 Sentiment Analysis Algorithms

At this point, the project has worked with four algorithms for analyzing sentiment in the Eli reviews. Three are implemented as UIMA annotators, in each case as the final primitive annotator in a sequence of annotators that also includes basic NLP functions such as tokenizing and sentence boundary detection. Another, experimental algorithm has only been implemented in a standalone program, because it was deemed not worth additional effort at this stage.

4.2.1 Bag-of-Words Analyzer

The first algorithm is a relatively straightforward machine-trained *bag-of-words* analyzer. A bag-of-words algorithm does not make use of any linguistic structure: it simply considers individual words. The Eli bag-of-words algorithm compared each word in the review text with predefined lists of terms that it associated with positive-sentiment and negative-sentiment reviews. For each match, it incremented an associated total-sentiment variable by the probability that the found word belongs to the positive or negative class. Finally, it would determine overall sentiment by comparing the positive and negative totals.

The words in the lists and their associated probabilities were assigned using a separate training program. The training algorithm extracted all the unique words from the 200 training documents and counted their occurrence in each of the two groups (positive and negative) of documents. Then words that occurred only a few times were removed from the lists, as were words that occurred about as frequently in both lists (normalized by the total number of words in each list). The resulting lists represent terms likely to appear in either a positive or a negative review, based on the training data.

It's worth noting that these lists do not in general correspond to any literal expression of sentiment. For example, terms in the negative list include "outline", "start", and "other"; the positive list has "argument" and "maybe". These are simply terms that students writing the reviews used in training tend to use when creating comments that expressed negative or positive sentiment.

In order to distinguish significant sentiment from neutral documents that simply contain a few terms from the lists, the bag-of-words analyzer applies a threshold value; calculated sentiment must exceed the threshold in order to appear in the results. The analyzer also requires that one of the sentiment features (positive or negative) exceed the other by a predefined margin, or it evaluates the document as having mixed sentiment. Currently, the threshold value is 3 sentiment-related terms, and the margin is a factor of 1.5; these are arbitrary values I selected intuitively which appear to give reasonably useful results.

Sentence-Mode Analysis. The bag-of-words analyzer has one other feature, which actually does add some consideration of linguistic structure: sentence-mode analysis. When this option is enabled, the analyzer processes the document sentence by sentence, rather than simply word by word. Within each sentence, sentiment-associated words are recognized, as per normal. However, at the end of each sentence, determination is made about the sentiment of that sentence based on the words found and the threshold and margin values. After all sentences have been processed, the document's sentiment is based on the relative counts of positive and negative sentences. (At this point the threshold is zero, so the document will not be considered neutral unless no sentiment-bearing sentences were identified.)

I introduced sentence-mode analysis on the intuition that in this genre, authors generally try to confine each sentence to one idea or a few related ones, and so sentiment will generally be consistent in a sentence. Thus sentence-level granularity of sentiment detection helps reduce noise. In practice, sentence-mode analysis did give slightly better results in testing, as noted below.

4.2.2 Trigram Analyzer

The second analysis algorithm attempts to incorporate some local context and provide better identification of sentiment-bearing phrases by incorporating *bigrams* (adjacent word pairs) and *trigrams* (triples) into the recognized-terms list. Trigrams, bigrams, and individual words or *unigrams*, collectively known as *n-grams*, are learned from the training sets of positive- and negative-sentiment reviews; then the analyzer scans input review texts for matching n-grams. This algorithm also uses the maximum likelihood estimation (MLE) of an n-gram to weigh the n-gram's contribution to the current total positive or negative sentiment. The MLE is the number of times the n-gram was observed in the training data, divided by the total number of n-grams of that order (i.e., the total number of tri-, bi-, or unigrams). So if the phrase "good job" appears frequently in positive-sentiment training data, for example, when that bigram appears in an input review it will contribute a relatively large amount to the estimated positive sentiment for that text.

There are various ways to combine the MLE weightings for n-grams of different orders. I won't go into details here, but the current implementation of the trigram analyzer offers a couple of simplistic combining modes, neither of which gives very good results in the testing done to date.

Like the bag-of-words analyzer, the trigram analyzer offers a sentence-oriented mode. It also has a number of other configuration parameters, such as a "sensitivity" setting that affects the threshold and margin values (which serve the same role as they did for the bag-of-words analyzer) and values that control how the n-gram weights are combined.

While the language model for the bag-of-words analyzer was trained using a separate utility, the UIMA implementation of the trigram analyzer includes its own training mode. This is of interest mostly to researchers developing other UIMA applications which need a training step before performing their normal processing.

4.2.3 Experimental Bigram Hidden Markov Model Analyzer

I experimented briefly with another n-gram analyzer. This one only uses bigrams and unigrams, but instead of simply adding weights for recognized n-grams, it uses a *hidden Markov model* and the *Viterbi algorithm* to find the most probable assignment of *sentiment tags* for a sentence. Here a sentiment tag is one of five values associated with each word: strongly negative, weakly negative, neutral, weakly positive, or strongly positive. The tag does not necessarily refer to the word itself, but to the change in sentiment since the last sentiment-bearing expression—for example, in a phrase like "your introduction is not as good as it might be", the word "might" could be tagged weakly negative, whereas the word "should" in the same position might merit a strongly-negative tag.

The bigram HMM analyzer is actually a standard HMM part-of-speech labeling algorithm, with sentiment tags substituting for parts of speech.

Unfortunately, getting decent results from this approach requires training it with a relatively large amount (on the order of thousands of sentences) of data which has been manually-annotated with sentiment tags by a human judge. This proved infeasible during the course of this stage of the project. It's possible that for another stage we'll develop a protocol for making this work more convenient and distributing it among a number of human judges.

4.2.4 Phrase-Based Analyzer

The fourth algorithm being explored for this project is still under development at the time of this writing, so no results are available yet. It tries to make more sophisticated use of linguistic structure, an important issue for sentiment estimation—the bag-of-words model often results in false-positive results because it finds a term that is often sentiment-bearing, but in a context that removes the sentiment, inverts its *polarity* (i.e., changes it from positive to negative or *vice versa*), or applies to something other than the presumed subject of the text. The local context used by the trigram model is only good for short phrases (e.g. "not very good"

as a negative, rather than positive, sentiment expression); human readers can detect subtleties of sentiment that are expressed by complex components of sentence structure.

The phrase-based analyzer is also of interest to researchers working with open-source NLP tools because it integrates UIMA with components of the OpenNLP project. Its UIMA application uses a tokenizer, sentence delimiter, part-of-speech tagger, and sentence parser from OpenNLP (and their associated data types), plus a token-filtering module and the actual sentiment analysis module written by me. As such, it's a useful example of how these disparate NLP technologies can easily be combined using UIMA.

This analyzer also differs from the others in that, rather than using a machine-learning approach, it constructs its language model from dictionaries compiled by other NLP projects. From the MPQA project [17] it gets a dictionary of sentiment-bearing words; these are labeled as strongly or weakly expressing sentiment, so the analyzer records that information too and uses it as a weighting factor. From the General Inquirer project[1] it gets a list of words that negate the sense of a phrase, and another of ones that decrease the sense; these are used to detect polarity reversal, as explained below. (Sense-decreasing words are considered polarity reversers because of their use in phrases like "low quality", which is a negative-sentiment expression.)

The phrase-based analyzer works this way: it splits the input text into sentences, then operates on each sentence in turn. It parses the sentences and searches the leaves of the parse tree (representing individual words or in some cases short phrases) for sentiment-bearing terms. If it finds any, it checks the immediate parent of that leaf—which will typically be a noun phrase or similar construct—for a reversal term. Then, if it has a result (positive or negative, strong or weak, possibly reversed from the simple sense of the word) for that leaf, it adds it to the total sentiment for the sentence. This design was inspired by the work of Nakagawa *et al.* [11], though it is far less sophisticated.

This is still a simplistic language model, but it does have the opportunity to consider some less-localized context, and specifically sentiment polarity reversal, by associating each sentiment-bearing term with the phrase it appears in. It also makes use of human-generated dictionaries, which may prove more accurate than the simple models generated from the relatively small amount of Eli training data.

5. RESULTS

At this point in the project, the results can be summed up as sufficiently interesting to warrant further investigation, but not yet useful for direct research application. The bag-of-words analyzer, which was originally intended as a baseline to compare more sophisticated algorithms against, currently offers the best results; and even those are not terribly good. However, these are still very preliminary given the small set of gold-standard data (N=50) I have to compare algorithm output against.

5.1 Algorithm Accuracy

Of the four algorithms described above, only the bag-of-words and trigram analyzers are in a state to produce results which can be compared against the gold standard. For that comparison I use the F_1 metric, a measure of accuracy. It combines *precision*, which is the proportion of correct results among document-sentiment labels assigned by the algorithm, and *recall*, which compares the number of correct results with the number of labels it should have found (i.e., the number of sentiment labels that are assigned to documents in the gold standard). Formally, F_1 is the harmonic mean of precision and recall, so it considers precision and recall as equally important. I look at precision and recall both jointly (in which case the sentiment determined by the algorithm must match the one I assigned manually), and independently; in the latter case, when considering positive sentiment, a result of "positive" from the analyzer is considered a match if the gold standard value is "mixed", for example.

	word mode	sentence mode
joint sentiment	0.62	0.69
positive only	0.76	0.79
negative only	0.69	0.64

Table 2: F_1 results for Bag-of-Words

	word mode	sentence mode
joint sentiment	0.53	0.60
positive only	0.75	0.71
negative only	0.45	0.51

Table 3: F_1 results for Trigram

The F_1 score ranges from 0 to 1, with the latter indicating perfect agreement with the gold standard. In order to make practical use of sentiment detection in Eli—for example, to extract a corpus of negative-sentiment reviews from a large Eli database for further analysis—we would probably want F_1 scores above 0.8 or possibly even higher, depending on our purpose. As you will see, none of the analyzers currently performs that well.

5.1.1 Bag-of-Words

In its default configuration, the bag-of-words analyzer produced F_1values between 0.62 and 0.76, depending on whether we're considering joint or independent positive/negative features. When run in sentence mode, the F_1 value for negative sentiment is somewhat lower, but overall that configuration returns better values (2).

5.1.2 Trigram

I tested the trigram analyzer in a number of configurations. Two did return results which approached those of the bag-of-words, though the trigram analyzer never did quite as well. Those were the trigram analyzer using interpolation (with default λ_n values) and sensitivity set close to the maximum (implying very low threshold and margin values), in word-by-word and sentence-granularity modes (3).

As you can see in the table, this analyzer did particularly poorly at identifying negative sentiment. I believe this is an artifact of the implementation, triggered by the *sparse data problem*: because the training set is small, the algorithm

recognizes very few complete trigrams or even bigrams in the input data. That causes it to put excessive weight on unigrams, which leads to numerous false positives. Then, because the implementation only looks for negative sentiment if it did not find positive sentiment, the false positives accumulate primarily on the positive side of the score.

5.1.3 Bigram HMM and Phrase

Experiments with the bigram HMM algorithm suggest that it might give results at least as good as the bag-of-words, given sufficient training data, which is what we would expect. Unfortunately, creating that training data would be a laborious process, so as noted above if we decide to pursue this approach we will need to provide a better system for doing so.

However, local-context approaches like the n-gram algorithms can never catch larger linguistic structures that affect sentiment, like longer phrases with reversal terms. For that reason, I believe it might be better to devote resources to approaches like the phrase analyzer. Preliminary results for that algorithm should be available soon, and if they are encouraging, it would be worth pursuing more sophisticated versions.

5.2 Algorithm Performance

One issue is the run-time performance of the analyzers: CPU, memory, and storage resources; time to initialize the system (often significant, since language models and similar resources need to be loaded, configuration must be processed, etc); and time to process each review document.

While the UIMA framework is very useful and greatly reduces the time and effort required to implement these algorithms, it does add significant overhead. A typical NLP primitive operation such as part-of-speech tagging might run a couple of orders of magnitude more slowly in a UIMA implementation, all else being equal. Similarly, while OpenNLP is very powerful, it is not particularly fast at operations such as parsing.

In my testing, running these algorithms on a relatively recent and powerful PC, the bag-of-words and trigram analyzers are fast enough to be used for on-demand sentiment determination of a handful of reviews. That might be useful for doing *ad hoc* visualizations and similar exploratory research in Eli. The phrase analyzer, on the other hand, is too slow, particularly during initialization, to be comfortably used interactively. It would be more effective for offline analysis, with results stored in the database and used later by researchers or instructors.

5.3 Significant Problems

5.3.1 Training Data

The training sets used for the machine-trained algorithms (bag-of-words and trigram) are too small, leading to sparse data issues, as noted above. Also, the training sets were created based on the numeric review ratings. That assumes that sentiment correlates to numeric ratings, which is unproven, and in fact is a hypothesis which we'd like to be able to test, but obviously cannot with a system trained under that assumption.

The bigram HMM algorithm requires a large amount of human-annotated data which is laborious to produce, and if produced by a single human judge may incorporate an unacceptable degree of a single subjective interpretation of sentiment.

The phrase algorithm builds its model using human-generated dictionaries, which avoids some of the problems with machine learning. However, it also restricts the amount of information available in the language model. It's also unknown at this point how well the word lists extracted from those dictionaries correspond to the diction employed by the student reviewers.

5.3.2 Sentiment Model

The representation of sentiment I used in this project, though a bit more complex than simple positive/negative, is still highly reductive. It fails to capture the wide range of sentiment that might be expressed even in a relatively narrow genre like writing peer reviews—affective responses such as "interesting" or "surprising", for example.

Also, reporting simply a single sentiment value for each document does not represent the common mix of sentiment in reviews, where the reviewer tries to describe both positive and negative aspects of the piece being reviewed.

5.3.3 Parameters

All of the algorithms except the HMM make use of rather arbitrary formulas with parameters that I chose intuitively or by trial and error, such as the threshold value and margin coefficient. Those parameters should be determined more rigorously, perhaps through Monte Carlo trials or, better, using an optimization approach such as Expectation Maximization.

6. NEXT STEPS

Finally, some brief remarks on the future of the project. First, of course, the phrase analyzer needs to be completed and tested, probably with some tuning work to experiment with various parameter values. That will help guide future research in this area.

The UIMA annotators are currently being run under UIMA utilities, which are intended for development. To be used in real applications, they need to be packaged and deployed with a complete run-time environment, their model data, etc. Then we will need to explore options for integration with Eli, probably through a loosely-coupled web-service architecture.

At that point the project will probably fork into two related efforts. One will be to improve Eli integration, including creating appropriate additions to the Eli user interface, and then from there into providing a general framework for doing natural-language processing of whatever sort on Eli-hosted data. The other will be to investigate sentiment analysis further, and in particular the relationship between sentiment and rhetorical tone.

7. ACKNOWLEDGMENTS

This project was developed for Joyce Chai's CSE841 Natural Language Processing class at Michigan State University, Spring 2011. The original Eli research project was developed by Mike Mcleod and Bill Hart-Davidson at the WIDE (Writing in Digital Environments) Research Center at Michigan State University. WIDE subsequently entered a partnership with Red Cedar Solutions Group in Okemos, Michigan, to produce a commercial version of Eli, which is currently in beta-test stage.

I'd like to thank Dr Chai for the suggestion which led to the project in its current formulation, for her advice and encouragement, and for an interesting and enlightening course. I'd also like to thank Bill Hart-Davidson and Mike Mcleod for access to the Eli system and data, and for numerous stimulating conversations about computational rhetoric in general and student writing-review evaluation specifically; this project would not exist without them. The generosity of researchers in NLP and related fields in making resources like UIMA, OpenNLP, MPQA, and General Inquirer available was also indispensable.

As always, my gratitude to Malea Powell for her encouragement and support.

8. REFERENCES

[1] General inquirer home page. http://www.wjh.harvard.edu/~inquirer/.

[2] Andreevskaia, A., and Bergler, S. Sentiment tagging of adjectives at the meaning level. In *Advances in Artificial Intelligence*. 2006, pp. 336–346.

[3] Andreevskaia, A., Bergler, S., and Urseanu, M. All blogs are not made equal: Exploring genre differences in sentiment tagging of blogs. In *International Conference on Weblogs and Social Media*. 2007.

[4] Apache Foundation. Apache uima. http://uima.apache.org/, 2011.

[5] Arora, S., Mayfield, E., Penstein-Rosé, C., and Nyberg, E. Sentiment classification using automatically extracted subgraph features. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text* (2010), pp. 131–139.

[6] Brody, S., and Elhadad, N. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010), pp. 804–812.

[7] Gamon, M., and Aue, A. Automatic identification of sentiment vocabulary: exploiting low association with known sentiment terms. In *Proceedings of the ACL 2005 Workshop on Feature Engineering for Machine Learning in NLP, ACL* (2005), pp. 57–64.

[8] Greene, S., and Resnik, P. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of Human Language Technologies: The 2009 Annual Conference ofthe North American Chapter of the Association for Computational Linguistics* (2009), pp. 503–511.

[9] Hart-Davidson, W., McLeod, M., Klerkx, C., and Wojcik, M. A method for measuring helpfulness in online peer review. In *Proceedings of the 28th ACM International Conference on Design of Communication - SIGDOC '10* (São Carlos, São Paulo, Brazil, 2010), p. 115.

[10] Liu, B. Sentiment analysis. *Invited talk at the 5th Annual Text Analytics Summit* (2009).

[11] Nakagawa, T., Inui, K., and Kurohashi, S. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2010), pp. 786–794.

[12] Pearson, T. How to build your own "watson jr." in your basement. https://www.ibm.com/developerworks/mydeveloperworks/blogs/InsideSystemStorage/entry/ibm_watson_how_to_build_your_own_watson_jr_in_your_basement7?lang=en, February 2011.

[13] Taboada, M., Brooke, J., and Stede, M. Genre-based paragraph classification for sentiment analysis. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (2009), pp. 62–70.

[14] Wei, W., and Gulla, J. A. Sentiment learning on product reviews via sentiment ontology tree. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (2010), pp. 404–413.

[15] WIDE. Eli development blog. http://eli.wide.msu.edu, 2011a.

[16] WIDE. Writing in digital environments (wide) research center. http://wide.msu.edu, 2011b.

[17] Wiebe, J. MPQA releases - corpus and opinion recognition system. http://www.cs.pitt.edu/mpqa/, 2002.

[18] Wojcik, M. Estimating ethos. http://www.ideoplast.org/rw/portfolio-2007-2008/ee/index.html, 2009.

[19] Wojcik, M. Inventing computational rhetoric. Presentation at the 62nd annual convention of the Conference on College Composition and Communication., April 2011.

[20] Wojcik, M. Sentiment analysis of student reviews of student writing. http://www.ideoplast.org/cse842/wojcik-sentiment.pdf, 2011.