IS256 Applied Natural Language Processing

Prof. Marti Hearst

UC Berkeley, School of Information

Fall 2006

We the News

Investigating Blog Punditry

Yiming Liu, Kevin Mateo Lim, Olga Amuzinskaya

TABLE OF CONTENTS

1. We the News

1.1 Introduction

We the News is conceived of as a project to examine, analyze, and classify blogosphere reactions to some controversial news event. Blogs are interesting in many ways. But sometimes they are interesting not for their “truth value,” but because they are personal and opinionated. We attempt to collect and classify blog entries as a gauge of public reaction toward news events, with the intent of automating the process by which core arguments and positions on a given topic can be identified and presented.

1.2 Project Goals

The project set out to build two primary components: a web service framework that allows semi real-time retrieval of news stories and blog entries that relate to those stories, and a NLP tool that uses the structured output from the retrieval service to rank, classify, and extract highly affective, charged statements from blog texts.

1.3 Conceptual Vision

The retrieval web service is a logistical framework that uses multiple public web service APIs and RSS feeds to obtain blog entries that relate to a specific event, whether extracted from a news service, or by the user manually typing in search parameters. In addition to API service adaptors and XML input and output processing modules, the service possesses a text extraction module that uses some predefined heuristics to identify and extract the blog full-text body, given a link to a HTML blog post.

The affect/opinion extraction and classification toolkit is of more interest in NLP terms. The classification tool extracts highly opinionated statements, given structured XML input of a set of blog entries. These statements can be considered “sound-bite” summaries of the entry in question, and should be in some sense very revealing of the blog author’s core opinions regarding the subject in question. The toolkit also contains a set of three human retrieved and constructed training corpora containing 20 to 30 blog posts each that correspond to a central topic of controversy.

We make a fundamental **hypothesis** that highly affective sentences are proxies for highly opinionated statements, and thus attempt to identify and extract these charged statements from the blog text. While this is not always the case, we assume that this is a reasonable approximation for the majority of the cases where we seek toidentify sentences that are revealing of core opinion. We assess the performance of our system, with respect to this hypothesis, in our discussion of results.

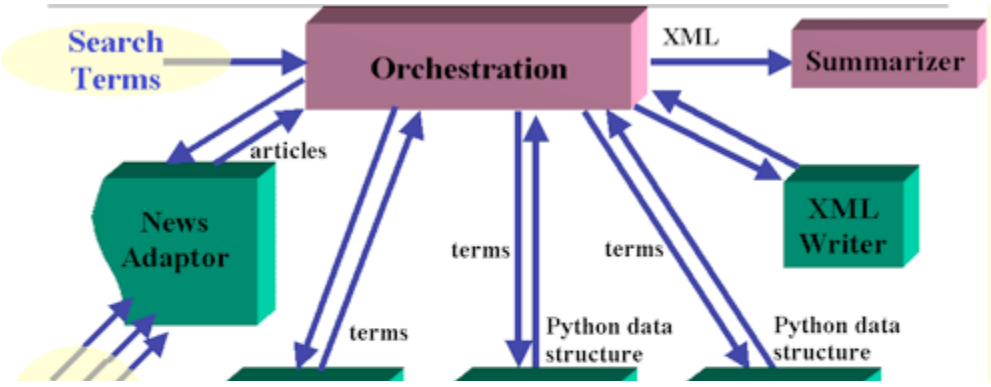
1.4 Task Breakdown

Team members held weekly meetings to coordinate project direction. Further, individual tasks were allocated as follows:

- Yiming - concept development, blog full-text extraction, SimpleClassifier, SummaryEvaluator
- Kevin - web service UI, NaiveBayesClassifier, Inquirer and general feature scoring
- Olga - XML schema design, XML Reader/Writer, feature scoring
- All members - training corpus development, classifier tuning and testing

2. Retrieval Service

2.1 Retrieval Architecture



The Orchestration.py module (implemented in Python) controls the web service that retrieves blogs posts (along with Flickr pictures) relevant to the news story. The orch.py file implements an object of an Orchestration class. blogservice.php is a thin PHP code layer to enable its deployment as a web service.

Parameters are passed to blogservice.php as HTTP GET or HTTP POST variables. They include:

- The kNumStoriesParam sets the limit to the number of new articles to retrieve. The default setting is 5.
- The kNumBlogsParam limits the number of blogs for each news event. The default is 5.
- The kNumTagsParam limits the number of tags (or term key words that characterize each story) to extract from each of the news stories. The tags extracted from each news story will be used as the search terms to find the blog posts around the story.
- The kTermsParam provides a manually created list of terms to use when searching for blogs instead of the tags extracted from each news story.
- The staticMode parameter is an override control to use a static set of prebuilt XML files rather than doing a live retrieval.
    - election – process news and blogs reactions to the recent elections
    - truth – process the news and blogs about the movie “Inconvenient Truth”
    - internet – process data about the Internet Explorer 7

Orchestration class uses the following data fields to store the extracted contents:

- collection field is an object of class NewsCollection defined within the Orchestration.py. It contains the dictionary of newsAtributes (title, source, date, authorship attributes of a collection of news articles about specific event) and the list of articles on this event. The articles is the collection of the objects of ArticleItem type. Each ArticleItem

contains fields such as title, link, summary, and the collections of photos, tags, and blogs associated with this news article.

- collectionXML is a string that contains a serialized collection datafield. It is used to output the results of news, photos, and blog extraction work performed by the Orchestration module.

The collection is filled when Orchestration utilized the following modules to implement its main functionality:

- News Adaptor module is implemented as a newsFeeder data field of the Orchestration class. Responsible for extracting either live RSS feed articles from specified live feed source or from a sample RSS feed stored in a static file (for testing purposes). Its class type is ArticleExtractor and it is defined in the ArticleExtractor.py.
- Term Extractor module is implemented as a termExtractor data field of Orchestation class. Responsible for extracting key terms from the text of a news story. It is of the type TermExtractor, a class defined in TermExtractor.py.
- Blog Adaptor is implemented as a blogExtractor field of the Orchestration class. Responsible for extracting full text blogs using blog post URLs from the Technorati API. Its class type is TechnoratiExtractor and it is implemented in the TechnoratiExtractor.py file.
- Photos Adaptor is a PhotoExtractor object implemented by PhotoExtractor.py file and its job is to find Flickr photos by using the search terms extracted from news articles.PhotoExtractor.py file and its job is to find Flickr photos by using the search terms extracted from news articles.
- XML Writer is a module module that serializes the collection of Attributes and ArticleItems into an XML text.

3. NLP Analysis Tool

3.1 NLP Analyzer Architecture

The NLP Analyzer consists of two Classifiers, supported by a number of feature scorers. Both classifiers operate with the NewsCollection data structure - they take in a NewsCollection as input, and outputs the same NewsCollection structure but with the necessary <emotionalSummary> fields filled with what they believe to be highly opinionated sentences.

- To facilitate easy operation on the NewsCollection structure, we provide two serialization classes. XMLReader will accept a document that conforms to our NewsCollection XML Schema (as defined in allNewsCollection.xsd) and deserialize it into a NewsCollection object. XMLWriter, in turn, will take a NewsCollection object and serialize it into NewsCollection XML Schema form.
- Scorers are used to identify the various features that constitute an effective opinion statement. These scorers are explored in detail this section.
- SimpleClassifier uses feature scores from all available scorers and implements a simple additive algorithm that sums total feature counts and selects the top ranking sentences for extraction into an "emotional summary", which is a set of sentences that represent the core opinion of the blog post being analyzed. The SimpleClassifier functions as a baseline, and is unsupervised.
- NaiveBayesClassifier implements a classical multinomial Naive Bayes classification algorithm to distinguish, rank, and extract sentences. It is a supervised method that requires a training collection of blog posts.

Both Classifiers are designed with Cross Validation and Testing modes. When run on the command line, each *Classifier.py module will execute a self-diagnostic, cross-validation routine. The threshold for cross-validation selection is 0.2 by default, which means 20 percent of posts will be selected as test posts. The topic corpus used for evaluation defaults to kElection, the election 2006 corpus. Classifiers accept a topic argument, however, which can be "ie" or "truth", to switch to a different corpus.

For Testing mode, the ClassifierTest.py driver script will accept a topic argument and use its corresponding corpus for training. The classifier will then be run on the testing corpus, a collection of about 10 posts for each

topic that have not been preprocessed by humans for the emotionalSummary.

SummaryEvaluator is a class designed to assess the performance of the Classifiers during cross-validation. In essence, it compares the original NewsCollection data structure as deserialized from the human-prepared gold standard XML files to the newly generated NewsCollection data structure from the Classifier. Discrepancies between each blog posts' extracted sentences and the extracted sentences in the gold standard are noted. The Evaluator will then generate a recall and precision report for the validation run, with some additional miscellaneous statistics.

3.2 Analyzer Modules

3.2.1 Feature Scorers

Each feature scorer that we implemented followed this convention:

- Implement method getScores()
  - Input 1: collection, type: newsCollection, this data structure that holds all the blog post texts for scoring
  - Input 2: topic, type: string, this contains the path to the XML file that should contain the topic-specific blog data. This parameter both indicates what topic and points to the correct data file. In practice, these were populated as the static values defined in the ServiceConfig module, e.g. kElection or kInconvenientTruth
  - Output: a 3d list of feature scores for the collection, where scores[i][j][k] is the feature score for the kth sentence of the jth blog post of the ith article in the collection
- Allow testing by hand with main method: Each method can be run in standalone (e.g. python curseWordScorer.py), in which the default topic is kElection (Democratic victory in 2006 election)

Scorers: Technique description, feature rationales

- **Strong Valence Word Scorer** - This scorer finds "strong" words based on the ANEW list of words. Since the ANEW list provides a valence score (1.0 = Most Extreme Negative, 9.0 = Most Extreme Positive), this scorer defines "strength" based on distance from the center of the continuum, 5.0. For a given sentence, this scorer will provide the number of words that are greater than 2 valence points from the center (i.e. words with

valence greater than 7.0 or less than 3.0).

The rationale here is that a sentence with strongly positive words (such as "affection," valence score 8.39) and strongly negative words (such as "whore," valence score 1.61) are more likely to be strongly emotional than sentences without these words.

- **Curse Word Scorer** - This scorer builds a dictionary of common curse words, as found here. Since stemming proved too slow over our corpora (see below), we defined two dictionaries for detecting curse words:
  - Curse words that should be detected as substring: certain curse words tend to be found within other words, which themselves we count as curse words. For example, substring "damn" is found in "goddamn" and "damnit". We deemed that, for certain cursewords, any word that contained one of these curses a substring was most likely a curse itself. It is unlikely that a word would have "damn" as a substring without evoking its profanity.
  - Curse words that should only be detected as a fullword: The risk of detecting any word that contains a curse is that words like "Massachusetts" would count for having "ass" as a substring. For this, we define a dictionary of words that must appear standing alone. However, understanding that words like "ass" have muliple incarnations as curses, we spelled those out (e.g. "asses", etc)

  We decided that sentences that included cursing were likely to be highly charged sentences: "Why must that ######### Hilary be so #######?"

- **pronounWordScorer** - This scorer builda a dictionary of pronouns, including their various forms. It then uses the dictionary to find any tokens within each sentence that are pronouns. The score count of a sentence is incremented per each token that is a pronoun.
- **imperativeSentenceScorer** - This scorer builds a dictionary of verbs, as found here. The scorer examines each sentence and increments score counts for those sentences that begin with a verb.
- **exclamationPointScorer** - Examines sentences and increments score counts for those sentences that end with an

exclamantion point.

- **positionSentenceScorer** - Identifies the position of each sentence in a body of a blog text and increments a score count for the sentences which are located either at the beginning or at the end of a blog.
- **SearchTermScorer** - Since both human input and automatic newsfeed extraction will yield a number of search terms or tags (the retrieval framework requires these tags to search for relevant blogs and photos), it is useful and simple to use these terms as cues for sentence extraction. Since we wish identify the opinion of the blogger with respect to some topic, it is likely that he or she will mention these topic words in the sentences where his or her opinion is given. For example, opinions regarding IE7 may likely refer to its topic, and therefore contain terms such as "IE7", "Internet", "Explorer", "Microsoft", etc. This feature scorer counts the number of times topical search terms or tags were used in a given sentence and weights those sentences accordingly.
- **General Inquirer Scorers**: These scorers are based on the General Inquirer (GI) ord list, which lists a long list of words with various tags. Each tag in the GI word set corresponds to a specific property. What we did is process the raw Inquirer data, stored as inqtabs.txt (tab delimited file), and for each tag, serialized a dictionary of all words bearing that tag. For example, in the serialized metadictionary for GI tags, (which we pickled as inquirer_dict.pkl), each tagname (e.g. 'Strong') is a key which yields a dictionary of words. These dictionaries were built using the inqtabs.txt raw data file, and the InquirerList.py module.

  So for each inquirer scorer, we load up the appropriate dictionary and return the counts for each sentence in each blog post, for a given topic. Since the inquirer dictionary lists root words, and stemming is expensive, we decided to define a list of suffixes which would be applied to each dictionary word. For example, "disgusting" is not in the dictionary for inquirer, but by listing "ing" as a suffix, we will try affixing "ing" to each dictionary word in finding matches. This general set of behaviors with regard to counting words is implemented in inqGeneralScoring.py module.

  - **inqEMOTWordScorer** - 'EMOT' is a tag in the GI list. Words tagged with 'EMOT' are words pertaining to emotion, such as 'disgust' and 'bitter.' This scorer returns count for words (or derivatives of words) that are tagged

'EMOT.'

- **inqPleasurScorer** - 'Pleasur' is a tag in the Inquirer list. Words tagged with 'Pleasur' are words that indicate enjoyment, such as 'admire' and 'bliss'. This scorer returns count for words (or derivatives of words) that are tagged 'Pleasur.'

  We decided that sentences including pleasure words are more likely to be affective than those that don't.

- **inqPainScorer** - 'Pain' is a tag in the Inquirer list. Words tagged with 'Pain' are words that indicate suffering, such as such as 'frustrate' and 'dread'. This scorer returns count for words (or derivatives of words) that are tagged 'Pain.'

  We decided that sentences including pain words are more likely to be affective than those that don't.

- **inqStrongWordScorer** - 'Strong' is a tag in the Inquirer list. Words tagged with 'Strong' are words that indicate strength, such as 'affirm' and 'approve'. This scorer returns count for words (or derivatives of words) that are tagged 'Strong.'

  We decided that sentences including Strong words are more likely to be affective than those that don't.

- **Cue Word Scorers**: Cue words are terms that indicate themselves as more or less likely to be in a given class (in our case whether or not the sentence is in an affective sentence). For each term, we calculated a selection ratio : (number of emotional summary sentences that contain the term) / (total number of sentences that contain this term). Stigma words we defined as words that had particularly low selection ratio (they seem less likely than average to be in the emotional sets we picked). Bonus words we defined as words that had a particularly high selection ratio (that is, they're more likely to be in the emotional sets we hand-picked). We selected the specific cue words for each domain using CueWordList.py. This generated a bonus and a stigma list for each domain. For a given domain, we stored the dictionaries as a pickled metadictionary with exactly two entries: 'bonus' was a key that mapped to a dictionary holding all the bonus words for the domain, 'stigma' mapped to a dictionary containing the stigma

terms. The files containing those serialized metadictionaries are in [topic]CueWords.pkl, where topic is in [InconvenientTruth, InternetExplorer, and Election]. We defined not a specific threshold value, but aimed to have the dictionaries be 10% of the words counted. Therefore, we simply ranked all the words in a domain by selection ratio, and figured out which single term was right on the cusp of the 10% borders (e.g. in a corpus of 3100 unique terms, we would select out the 310th highest and 310th lowest terms as ranked by selection ratio. These two words defined our cutoff ratios.

We believed our domain-specific development sets to be good for finding good bonus words and stigma words for that type of blog post. For instance "scandal" is an especially charged word, given the domain of the 2006 American election results. Instead of relying on another authoritative list (such as ANEW or General Inquirer), we decided it would be a strength to use our own development sets to create a list of words. And in our cross validation trials, the Cue Word scorers benefitted both recall and precision.

- **StigmaCueWordScorer** - After creating the list of cue words, this scorer counts the number of stigma words in each sentence.

  We determined that words that showed up disproportionately little in the affective sentences of our development corpora would indicate lower affect in general.

- **BonusCueWordScorer** - After creating the list of cue words, this scorer counts the number of bonus words in each sentence.

  We determined that words that showed up a disproportionately high number of times in the affective sentences of our development corpora would indicate higher affect/charge in general.

- **CapWordsScorer** - Capitalized words, especial ALLCAPS PHRASES or key proper nouns such as "Democratic Party", will tend to be cues for useful sentence extraction. With the pseudo-formal styles of most blogs, capitalization is often a textual representation of shouting or emphasis. Further, capitalized words are typically unique named entities that are being discussed, which is often revealing of the blogger's

opinion. This feature scorer implements capitalization weights using both a simple additive algorithm and the Kupiec capitalization scoring method. It defaults to Kupiec mode, which grants 1 unit of weight to capitalized words that are not sentence initial, with additional weight granted to the sentence that contains the first occurrence of that particular word in the document.

Challenges in scorers:

- Slowness of stemming: in the scorers that built a dictionary for scoring, it would have been useful to normalize words into base forms for comparison. Unfortunately, Wordnet's morphy() reduced scorer performance **significantly**, which constituted a very problematic bottleneck. The Porter Stemmer is much faster, but did not allow us to normalize to actual words. Non-word roots could not compared to dictionary words that were contained in those aforementioned scorers, which made the entire exercise useless.

3.2.2 XML Reader

XML Reader receives an XML either as an input from the Orchestration module or reads in a static file (such as a Gold Standard blog collection for each of the sample news events). It extracts data about the new articles and relevant blogs, and produces a Python data structure with the datafields filled with the extracted data.

3.2.3 Simple Classifier

Simple classifier is implemented in SimpleClassifier.py module. It is an unsupervised method which does not do any self-training and uses all of the available feature scorers to select sentences with the highest feature count.

3.2.4 Naive Bayes Classifier

The classifier is used to classify each blog sentence as either a good represenative of the opinion of blog or not. It is called from the SummarizeTest.py file.

The NaiveBayesClassifier class requires the following input arguments to initialize:

- training collection data structure, representing a set of blogs with manually selected summary sentences

- topic to search for the test blogs to evaluate the effectiveness of the trained classifier
- threshhold is used to adjust the fraction of sentences that should be extracted from the full text to serve as summary.

The classifier is implemented in NaiveBayesClassifier.py file. It trains itself by assigning the higher weights to the scores which were most successful in identifying the correct summary sentences in the Gold Standard (the train() method). When the training is complete, it attempts to summarize the test set of blogs (the summarize() method) and uses the SummaryEvaluator.py in order to analyze how well the classifier was able to pick the emotional sentences from the test collection.

3.3 Training Corpora

To facilitate training and testing of the classifiers, three training corpora of blog posts were developed, ranging from 20 to 30 blog posts each. These posts pertain to the central theme of their respective corpora, which consists of three prominent news events:

- U.S. Election of 2006 - the Democratic Party captured majorities in the legislative branch. Bloggers of both sides of the political spectrum weighed in on the aftermath.
- "An Inconvenient Truth" - Al Gore's documentary on global warming provokes reaction in the blogosphere.
- The release of Microsoft Internet Explorer 7 - after years of development and much fanfare, Microsoft (an ever-controversial firm in the tech world) and its new Internet Explorer faces the amateur tech pundits of the WWW.

Each corpus is manually developed by a member of the project team, who also makes a qualitative judgment on the valence of the opinion of the blogger with respect to the topic on a 1-9 scale, 9 being most favorable and 1 being least favorable. The team member also selects a number of sentences which he or she believes is most reflective of the blog's core opinion (and thus the targets for automatic extraction). This metadata is inserted into the markup for use by the evaluator module.

4. Results

4.1 Context

Using the Simple Classifier as the baseline extraction tool and the Naive

Bayes Classifier as the main extraction tool, we performed cross-validation with all feature scorers on the three gold standard corpora. From each corpus, 20 percent of its blog posts are randomly designated as test posts, while the remaining 80 percent are designated training posts. The classifiers are then trained and tested on the respective posts. The SummaryEvaluator is then used to compare machine-selected sentences with human-selected sentences in the corpora, and precision/recall scores are output.

Because of the random-selection of posts used for cross-validation, we run each classifier 5 times and take the average recall and precision ratings for comparison.

Finally, we train the classifiers on a training corpus, run it on a separate test set of posts on the same subject, and qualitatively assess the accuracy.

4.2 Result Details

4.2.1 News Event 1: Election

Election Cross Validation Results

Classifier	Recall	Precision
Simple	0.32	0.30
NaiveBayes	0.71	0.79

Sample run:

Al Jazeera (and others) report that Islamic radicals in the middle-east are praising American voters for "defeating and rejecting Bush's failed policies" following yesterday's midterm elections. This is all that really needs to be said about yesterday's big Democrat victory(s), (UPDATE: but here's more)

But I can't resist digging deeper. **How miserable is your political party when you have the enemy of your country cheering for your victrory as a sign that they have won the "hearts and minds" of Americans? Terrorists are cheering because Democrats have been championing their cause since 2003 and they believe all of the Democrat rhetoric and benefit from it. (Is the above photo from Al Jazeera or a Democrat rally?) Islamic throat-cutting fascists know that a Democrat win is a win for Islamic throat-cutting fascists.** How so? They have been doing this for a while now. They believed Democrats when they said "this is a war America cannot win". They believe Democrats when they say America must get out of Iraq (cut and run) ASAP. They agree with John Kerry and think our troops are uneducated idiots. They agree with Democrats who

want to let them phone America without the CIA/NSA/FBI etc listening in on their calls. They love the fact that Democrats want to give them a free lawyer and full US citizenship rights if captured while trying to kill US troops. They repeat Democrat talking points (almost word for word) in their speeches when they say that America is the problem in the world and should concentrate on our own healthcare system and feeding the poor instead of the war on terror in Iraq.

What is this so-called New Direction? If siding with the enemy of America during a time of war is Democrats idea of a "New Direction" then "God help us". How can Democrats be proud when their victory is considered even more of a victory by America's enemy? (Islamic throat-cutting fascists) It's easy to understand why terrorists would support Democrats, but why would American voters? "Props" go to CNN, MSNBC, ABC News, CBS News, NBC News, Jon Stewart, David Letterman, Whoopie Goldberg, Sean Penn, Rosie O'Donnell, The Dixie Chicks, Bill Mahr, Al Franken, Michael Moore and the NY Times and virtually every other liberal newspaper.(All of whom are cheering along with the terrorists) Now, what happens if America doesn't live-up to Democrats promises to "cut and run"?

The problem is that liberals in America have eagerly let yourselves be used as "useful Idiots" in order to bring down Bush.The plan is to kill you along with the rest of us when the time comes. **Yesterday was a victory for all of you useful idiots who claim to be smarter than everyone else and a victory for the terrorists who played you like idiots against your own government.** Congratulations on your "victory". One other interesting note: Democrats and all the news media talking about nothing but rampant voter fraud and problems with the voting process for days and even on election day right up until the time Democrats started winning, then suddenly no further mention by anyone anywhere of ANY problems with the count. Hmmmmm Had Democrats lost, we would be knee-deep in lawyers right now. I'm just sayin'...

EXTRACTED: Terrorists are cheering because Democrats have been championing their cause since 2003 and they believe all of the Democrat rhetoric and benefit from it. (Is the above photo from Al Jazeera or a Democrat rally?) Islamic throat-cutting fascists know that a Democrat win is a win for Islamic throat-cutting fascists.

EXTRACTED: How miserable is your political party when you have the enemy of your country cheering for your victrory as a sign that they have won the "hearts and minds" of Americans?

MISSED: Yesterday was a victory for all of you useful idiots who claim to be smarter than everyone else and a victory for the terrorists who played you like idiots against your own government.

4.2.2 News Event 2: "An Inconvenient Truth"

Inconv. Truth Corpus Cross Validation Results

Classifier	Recall	Precision
Simple	0.15	017
NaiveBayes	0.55	0.73

4.2.3 News Event 3: Internet Explorer

Internet Explorer Corpus Cross Validation Results

Classifier	Recall	Precision
Simple	0.19	0.40
NaiveBayes	0.38	0.57

4.2.3 Test Set extraction run: Election

Trying To Savor The Taste... Current mood: relieved Current music: "Blow Northerne Wind"--The Mediaeval Baebes Trying To Savor The Taste... But I barely recognize it any more! We all know what triumph tastes like, but I'm having to get used to not feeling defeated and depressed after an election! I was sick as a dog on Wednesday morning--couldn't sleep Tuesday night, was nauseated half the day. Dunno what was up! After elections I'd worked, where I had often gone with less than three hours of sleep several nights running leading up to the big day, I would get what Mum called "Campaign Hangovers." The exhaustion (and usually the grief of the loss) would hit the morning after, I'd be dead to the world, often miss school , and usually come down with a head cold. Maybe Wednesday was a vicarious campaign hangover. Maybe I was feeling it sympathetically for all the stalwarts who still managed to trudge through that ghastly campaign after years of heartbreak. I salute them! We did it. My god, we did it. We took both houses. We got to see Rummy's scuzzy head on the wall! I'd never have thought it possible. I went to lunch with a colleague Wednesday who shares my political leanings, and in between talking shop, we pondered the election outcome and how Virginia and Montana (still up in the air at the time) would play out. As we were leaving the restaurant, there was a building across the street with a big TV screen and ticker on the side (a la Times Square, only about 1/10th the size). There it was, the headline: "RUMSFELD TO STEP DOWN!!!" I couldn't even speak; just squeaked like an idiot and pointed. By the time we got back to the office, Montana was ours. Last night, Virginia was projected for us. I've been alternating between fits of spontaneous, hysterical giggles and wanting to burst into tears of sheer relief. Another fellow Democrat at work (there's a surprisingly

large number of liberals at my firm--it might even be 50/50 which is unusual for a big office), remarked to me that we were having a "very subtle victory party" on Wednesday...which consisted of leaning in each other's doorways and hissing, "Didja see? Didja see? We got Rumsfeld/Montana/Virginia! Allen conceded! This is it! This is it!" and doing miniature happy-dances in our chairs. Such is politics in the professional world, I guess. We did it. O Joy, O Rapture, O Bliss. We actually did it. Well fought, my friends! Well sung and well danced! And to my Dems who may now consider themselves the Congressional Leadership...don't blow it. We need this to last until '08!

The exhaustion (and usually the grief of the loss) would hit the morning after, I'd be dead to the world, often miss school , and usually come down with a head cold. O Joy, O Rapture, O Bliss. There it was, the headline: "RUMSFELD TO STEP DOWN!!!" And to my Dems who may now consider themselves the Congressional Leadership...don't blow it. I've been alternating between fits of spontaneous, hysterical giggles and wanting to burst into tears of sheer relief. Maybe I was feeling it sympathetically for all the stalwarts who still managed to trudge through that ghastly campaign after years of heartbreak.

4.3 Discussion

We hypothesized that identifying highly opinionated statements can be approximated by identifying and extracting highly emotional sentences from the blog text. Overall, this has proven to be a reasonable hypothesis, especially in political or politicized domains where the event in question generates very charged statements, such as the election of 2006 and "An Inconvenient Truth". When bloggers emphasized their particular viewpoints, they often included some form of emphasis using loaded words, capitalization, punctuation, and other features that can be captured by our scorers. Instances such as "RUMSFELD TO STEP DOWN!!!" and "I've been alternating between fits of spontaneous, hysterical giggles and wanting to burst into tears of sheer relief" were very prominent examples of this.

This hypothesis has proven less accurate with the Internet Explorer event, which while having some emotion, is usually less well-defined, buried under sarcastic/bitter musings ("IE7 is clearly a Microsoft Product."), or consist of very factual, very frank comparison of technical features (such as IE 7 vs Firefox 2). In this latter instance, affect detection was not a very effectual approximation for opinion.

We also noted a phenomenon in which highly charged sentences were not core opinion statements in and of themselves, but serve as cues that indicate core opinion statements were in close proximity. In several instances, the sentences that were clearly opinion-summary statements preceded or followed after a very affectively charged sentence. We did not have enough time to extend our scoring framework to capture this particular feature, but this is a very interesting example of an extension to the current methodology in order to capture more accurately the desired statements.

The analysis tool generally achieved higher precision than recall for any given subject domain, implying that it was reasonably accurate at identifying opinion-summary statements but was not sufficiently sensitive to produce many of these statements from the general body of text.

Multinomial Naive Bayes classification outperformed the baseline on all measures, achieving some ~40 percent higher precision on most corpora, despite its assumptions regarding independence of feature probabilities.

5. Project challenges and future work

Challenges:

- Full-text extraction from arbitrary HTML blog pages has turned out to be a semi-NLP task in of itself. How should we identify the main body of blog text and extract them out of the sea of tags, navigation headers, irrelevant content (such as the "Current Mood" section in Livejournals)? In the absence of the Semantic Web, NLP techniques may be useful in creating a better blog crawler.
- Python multi-threaded execution is hobbled by the Global Interpreter Lock, which is basically a hack that makes Python thread-safe when it is not actually designed for multi-threaded execution. This has a problem where pure Python programs will not benefit significantly from implementing multi-threading, since Python threads will each be waiting sequentially for the interpreter lock to be released. In TechnoratiExtractor, our performance in live feed and blog retrieval is similarly hobbled by the threading abilities of Python. If this functionality were to be implemented in C, however, performance should improve significantly.
- Segmentation - currently, sentence segmentation operates on a basis of detecting punctuation and capitalization. However, blogs are an informal writing medium at times, and bloggers do not often use proper capitalization, newlines, or even punctuation. To generalize the segmenter to handle these instances would be another NLP project in of itself, but may improve the performance of the NLP analyzer tools when faced with these particular corner-case types of blog entries.

Future Work: the Mechanical Pundit

Ultimately, work in opinion detection, extraction, summarization, and affect scoring leads to an interesting possibility in building an advocacy agent. The distilled sentiments, opinions, and assertions of various bloggers can be stored in a database, along with metadata that extracts core opinions and evaluate them as points or counterpoints. Thus when queried on any topic of controversey, the agent may 1) inform the user as to the state of the debate, the sentiments involved, and arguments being presented, 2) play Devil's advocate in testing out arguments, and 3) become a pundit of at least equal caliber to late-night radio talk

show hosts. An agent such as this may be of some interest academically, but also as a kind of technology entertainment/art piece for others.

6. References

- Bradley, M.M., & Lang, P.J. (1999). Affective norms for English words (ANEW): Stimuli, instruction manual and affective ratings. Technical report C-1, Gainesville, FL. The Center for Research in Psychophysiology, University of Florida. http://www.phhp.ufl.edu/csea/Media.html
- General Inquirer - Tag Categories: http://www.wjh.harvard.edu/~inquirer/homecat.htm, Spreadsheets: http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm
- BeautifulSoup.py - an error-tolerant Python HTML parser. http://www.crummy.com/software/BeautifulSoup/
- J. Kupiec, J. Pedersen, F. Chen, A trainable document summarizer, Proc. of SIGIR, 1995.
- Wikipedia reference on profanity: http://en.wikipedia.org/wiki/Profanity
- Threading in Python - http://docs.python.org/api/threads.html
- Levin, Beth. Index from Engish Verb Classes And Alternations: A Preliminary Investigation. The Univ. of Chicago Press. http://www-personal.umich.edu/~jlawler/levin.html