Julie Kane Ahkter (kanej)
Steven Soria (ssoriajr)

Sentiment Analysis: Facebook Status Messages

Final Project CS224N

Abstract

While recent NLP-based sentiment analysis has centered around Twitter and product/service reviews, we believe it is possible to more accurately classify the emotion in Facebook status messages due to their nature. Facebook status messages are more succinct than reviews, and are easier to classify than tweets because their ability to contain more characters allows for better writing and a more accurate portrayal of emotions. We analyze the suitability of various approaches to Facebook status messages by comparing the performance of a Maximum Entropy ("MaxEnt") classifier, a MaxEnt classifier augmented with Labeled-LDA ("LDA") data, a MaxEnt classifier augmented with part-of-speech ("POS") tagging, and a MaxEnt classifier augmented with both LDA and POS data. We classify both binary and multi-class sentiment labeling. In both cases, a MaxEnt classifier augmented with POS data performs the best, achieving an average binary classification F1 score of approximately 85% and an average multi-class F1 score of approximately 67%.

Introduction/Background

With the transition from forum- and blog-based Internet communication among users to social networking sites such as Facebook and Twitter, there exists new opportunity for improved information mining via NLP sentiment analysis. Prior success at such analysis has been elusive due to the inherent difficulty in extracting a singleton sentiment label from long passages, where fluctuations over the document length make classification difficult. While sentiment analysis of Twitter data has surged in recent years, it too is problematic for the opposite reason: tweets are limited to 140 characters. Also, tweets often use heavy abbreviation and are more likely to be fragmented expressions, making it difficult to use a part-of-speech ("POS") tagger.

This project focuses on classifying sentiment of Facebook status updates ("status updates") using binary and multi-class labels. Facebook makes a distinction between a Facebook user's status update, versus links to a news article or other source of information, versus comments that are a response to another Facebook user. Unlike tweets, status updates can use up to 420 characters. Thus, status updates more often are written in mostly sentence-like structures that can benefit from POS analysis.

Why perform sentiment analysis on Facebook status updates? According to a January 2010 article on InsideFacebook.com, users spent nearly 7 hours per person on Facebook in December 2009, far higher than the other top 10 parent companies on the Internet. From a marketing standpoint, understanding user sentiment as it relates to a topic of interest clearly allows more effective ad-targeting. If a user is trending positively or negatively about health care reform, appropriate political party ads might appear sympathetic to a user's viewpoint. Similarly, a user that tends to write playful status messages might be shown ads for a local comedy club. Beyond the obvious marketing appeal, however, is a more sociologically compelling rationale: modeling the ebb and flow of consumer sentiment across a broad swath of topics, hierarchically encapsulated by user, locally-defined user communities, regionally, nationally, and globally.

To our knowledge, there exists no prior work that focuses exclusively on sentiment analysis of Facebook status messages. Nonetheless, we found (Ramage, Dumais, Liebling)'s work on augmenting a MaxEnt classifier with LDA data particularly insightful. While their work classifies tweets according to topic rather than sentiment, via an approach that extrapolates this from latent topics among the words in a tweet, the promise that a labeled LDA could combine the best of supervised learning while still discovering features in an unsupervised fashion seemed like a promising approach for modeling the inherent complexities of sentiment. Due to time constraints for this project, we chose to let the LDA tell us the ratio of words more closely associated with a fixed set of pre-labeled sentiments.

(O'Connor, Balasubramanyan, Routledge, Smith)'s was the first paper we read that shared a similar vision of NLP work of providing insight into public opinion by modeling sentiment within various societal clusters. Importantly, however, their work performs binary labeling against a pre-defined corpus of known positive/negative words. Our work aims to learn sentiment non-deterministically, training on nothing more than the sentence and a label for the sentence as a whole.

Data

To collect data, we created and registered a Facebook Connect application, called *iFeel*. Allowing the app to require specific Facebook privileges allowed us to host the app from our stanford.edu accounts, making it available to anyone, not just our friends, and allowing us to gather status updates quickly and efficiently. *iFeel* is a PHP-based app that uses the newer Facebook Graph APIs, which simply use URL-based queries to return result sets such as status updates. Around *iFeel*, we wrapped an HTML-based front-end that disclosed privacy and disclaimer information such as the period of retention of data, the usage of data, anonymity of data, project scope and purpose, et cetera. At the time of this writing, *iFeel* lives at http://www.stanford.edu/~ssoriajr, although this is subject to removal at any time.

iFeel collects the 25 most recent status updates from a logged-on Facebook user and their friends. Since Facebook users may have intersecting sets of friends, we performed a unique sort after merging all status updates to eliminate repeated status messages. Over a period of one week, we collected 62,202 unique, raw (unlabeled) Facebook status messages.

After collecting the data, the next step involved preprocessing the raw Facebook data, converting it into labeled data and breaking it up into training and test sets. For this project, we used the *Stanford Classifier* v2.0 (the MaxEnt classifier), *Stanford Tagger* v3.0 (2010-05-26, English-only) (the POS tagger), and the *Stanford Topic Modeling Toolbox* v0.2.1 (the Labeled LDA). All of these tools are available from http://www-nlp.stanford.edu/software/. Some massaging of the data between stages was required. Our high-level sequence involved the following:

1) Raw data collection via *iFeel*
2) Sentiment labeling
3) Transform into train/test sets for classifier-only tests
4) Transform into train/test sets for LDA
5) Transform into train/test sets for POS
6) Transform train/test sets for final classification by classifier
7) Adjust classifier, LDA, and/or POS features and repeat until best model

PREPROCESSING

Adapting a strategy used by (O'Connor, Balasubramanyan, Routledge, Smith), we labeled status messages by filtering for known emoticon equivalence classes, as defined by the Wikipedia article, *List of Emoticons*, at http://en.wikipedia.org/wiki/List_of_emoticons. We chose to filter for 14 categories: angry, evil, frown, rock-on, shock, smiley, wink, angel, blush, fail, neutral, shades, skeptical, and tongue. To prevent training on the very emoticons upon which we based our labeling, we then stripped these emoticons out of the data. For binary classification, we chose to classify into *positive* and *negative* sentiment labels. These labels are represented by:

POSITIVE:	smiley, wink, tongue, angel, shades, blush, rockon
NEGATIVE:	frown, shock, skeptical, evil, angry, fail

Certainly arguments can be made about our choices (is shock always negative?), but we chose to maximize our dataset size, given time constraints on data collection. After dividing the categories in this way and truncating *positive* and *negative* sets to ensure they are the same size, we ended up with 4,320 usable, labeled samples. We chose to split the data into roughly 75% training / 25% testing, resulting in a training set of 3,500 samples and testing set of 820 samples. *Positive* and *negative* samples were split into groups of 25 and then interleaved so that they were approximately evenly distributed within the training and test set files.

For the multi-class case, we chose to classify into four sentiment labels: *unhappy, happy, skeptical,* and *playful*, chosen to maximize the amount of usable data. These labels are represented by:

UNHAPPY:	evil, frown, shock, angry
HAPPY:	smiley
SKEPTICAL:	skeptical
PLAYFUL:	angel, rock-on, shades, tongue, wink

Again, truncating each class so that they are all equal, we end up with 3,612 usable, labeled samples spanning the classes above. Since we reasoned that the multi-class classification would be harder, we chose to favor training set size over test set size, splitting the data into roughly 90% training / 10% testing. This resulted in a training set of 3,300 samples and testing set of 312 samples. As before, each label was interleaved roughly evenly into the training and test files.

Next, we had to create versions of the training and test set files that were compatible with the tool to be used. After running POS and LDA against their respective training/test sets, we created Java programs to merge, in various ways commensurate with testing objectives, the resulting POS file of tagged status updates with the resulting LDA file of word-to-label ratios to produce a final training and test set to feed into the classifier.

Features

Based on previous experience, we expected n-gram word features to be the most significant such features, but also felt word-shape features would play a bigger role in this project, where sentiment polarity is often expressed by "!!", "??", "?!?!", "……", and so on. To this end, consistent across all our experiments, we split words in MaxEnt based on the regular expression, `[, ]` (comma, space). We expected that the degree to which we needed to tweak features between the binary and muli-label classification tasks would indicate whether Facebook messages truly encoded sentiment such that NLP models scale well to multiple sentiment labels.

TWO CLASSES: POSITIVE AND NEGATIVE

Our baseline results were determined by passing the training and test sets, without any POS or LDA data, directly into the classifier, using what we defined to be minimal n-gram based features. With those results, we adjusted classifier-specific features, such as word-shapes, observing any changes in the classifier's performance and saving the best feature set for the classifier. LDA features were more limited but the resulting word-to-label ratios became the LDA's best feature set and were combined with our classifier's best features. POS features were uniquely computed since the POS tagger simply output tagged versions of the training and test data files. Thus, we merged the LDA and POS results to count certain characteristics of the POS-tagged files such as number of nouns, verbs, adjectives, et cetera. Those real-valued counts and ratios, combined with LDA real-valued ratios, were further combined to produce new training and test files.

Specifically, our testing typically used the following combinations of data:

`fbs.binomial.train/test`	binary labeled data with no POS tagging or features
`fbs.binomial.LDA.train/test`	binary labeled data augmented with LDA features
`fbs.binomial.combined.train/test`	binary labeled data with POS tagged sentences and augmented with both LDA and POS features (could also be generated to contain POS tagged sentences but augmented with only POS features)
`fbs.mc.train/test`	multi-class labeled data with no POS tagging or features
`fbs.mc.LDA.train/test`	multi-class labeled data augmented with LDA features
`fbs.mc.combined.train/test`	multi-class labeled data with POS tagged sentences and augmented with both LDA and POS features (could also be generated to contain POS tagged sentences but augmented with only POS features)

CLASSIFIER

From past experience, we've seen that modeling n-grams are valuable features in a MaxEnt model and thus we set a baseline that enabled all built-in n-gram related features of the Stanford Classifier. The Stanford Classifier also allowed us to specify the split-word regular expression to use, and so we chose to filter on spaces and commas. Our baseline feature set for the classifier only consists of:

Stanford Classifier Baseline Features for Binary Classification

Property	Feature Name
useSplitWords	SW-*str*
useSplitWordPairs	SWP-*str1*-*str2*
useSplitFirstLastWords	SFW-*str* SLW-*str*
useSplitPrefixSuffixNGrams	S#B-*str* S#E-*str*

With this feature set to start with, our baseline F1 scores for the classifier were:

Stanford Classifier Baseline F1 Score for Binary Classification

Negative F1 Score	Positive F1 Score
0.759	0.737

Even with these high F1 scores, we observed several interesting examples of misclassification. Consider the following POSITIVE status update example: *is excited to officially have Fridays off work!! 3-day weekends and 4-day work weeks. Gotta love it!!* In our baseline run, this is classified as NEGATIVE. The problem here is the word *work*. Among the top 200 weighted features, the classifier reports this:

```
(1-SW-work,NEGATIVE)                        0.7427
```

Observe that there are two instances of *work* in the sentence. This classification is, therefore, not an unreasonable one. Similarly, *is excited* is also given a negative score of 0.33, despite the fact that both *is* and *excited* are learned as positive. The sentence is thus scored as being 74% likely to be negative. If we switch to using the Stanford Classifier's built-in word-shape feature set, "dan2", the sentence is correctly classified as positive.

Classification example: *is excited to officially have Fridays off work!! 3-day weekends and 4-day work weeks. Gotta love it!!*

Classifier with baseline features			Classifier with "dan2" features		
	NEGATIVE	POSITIVE		NEGATIVE	POSITIVE
…			…		
1-SW-work	0.74	-0.74	1-SSHAPE-WT-x!	-0.92	0.92
1-SFW-is	-0.08	0.08	1-SW-work	0.83	-0.83
1-SW-excited	-0.56	0.56	1-SFW-is	-0.19	0.19
1-SWP-is-excited	0.33	-0.33	1-SW-excited	-0.43	0.43
…			1-SWP-is-excited	0.20	-0.20
			…		
Prob:	0.74	0.26	Prob:	0.25	0.75

The "dan2" feature set creates features for word shapes with lowercase words, uppercase words, digits, mixed-case words, punctuated words, and equivalence classes words of the same shape with 3 characters or less. The key word-shape feature for this example is the *x!* shown above. This example has two words that end with a "!" character, which are now modeled. The overwhelming weight of *x!* as a positive influencer is enough to properly classify this example.

Still, "dan2" remains imperfect, especially in borderline cases and extremely short status updates (ala Twitter), as in the status update: *Disneyland tonight*. Switching to the built-in "chris4" feature set corrects this misclassification.

Classification example: *Disneyland tonight*

Classifier with "dan2" features			Classifier with "chris4" features		
	NEGATIVE	POSITIVE		NEGATIVE	POSITIVE
…			…		
1-SW-tonight	-0.47	0.47	1-SW-tonight	-0.45	0.45
1-SW-Disneyland	0.38	-0.38	1-SW-Disneyland	0.34	-0.34
1-SSHAPE-WT-x	-0.49	0.49	1-SSHAPE-xxxxx	-0.28	0.28
1-SSHAPE-WT-Xx	-0.30	0.30	1-SSHAPE-Xxxxx	-0.41	0.41
1-SFW-Disneyland	0.09	-0.09	1-SFW-Disneyland	0.07	-0.07
1-SWP-Disneyland-tonight	0.00	0.00	1-SWP-Disneyland-tonight	0.00	0.00
1-SLW-tonight	-0.10	0.10	1-SLW-tonight	-0.09	0.09
…			…		
Prob:	0.52	0.48	Prob:	0.40	0.60

The key feature introduced by "chris4" is that longer mixed-case words are learned to be positive. "chris4" also treats, as separate features, distinguishable, 2-character head and tail sequences. The interiors of words are equivalence classed similarly to "dan2" in terms of case-sensitivity, digits, punctuation, and others. This method also ensures that words ≤ 4 characters are modeled precisely. In this case, while *Disneyland* remains an apparently negative word according to the classifier, the equivalence class Xxxxx is positive, resulting in a 12 point shift of probability mass to the positive classification.

"chris4" addressed all the word-shape features we wanted to encode. In our testing, "chris2" performed identically to "chris4", but we ultimately chose to standardize on "chris4" since it handles Unicode characters better. Thus, our final feature set appears below. For more information on built-in word-shape feature sets of the Stanford Classifier, please refer to the Javadocs included in the distributable in stanford-classifier-2008-04-18/javadoc/index.html and select the *ColumnDataClassifier* class.

Stanford Classifier "Best" Features for Binary Classification

Property	Value	Feature Name

useClassFeature	true	n/a
1.splitWordsRegexp	[,]	n/a
1.useSplitWords	true	SW-*str*
1.useSplitWordPairs	true	SWP-*str1*-*str2*
1.useSplitFirstLastWords	true	SFW-*str* SLW-*str*
1.useSplitPrefixSuffixNGrams	true	S#B-*str* S#E-*str*
1.splitWordShape	chris4	SSHAPE-Xx…xX SSHAPE-xx…xx SSHAPE-dx…xd SSHAPE-Xd…xX SSHAPE-dX…s…+d SSHAPE-d-…/X …and so on

Note that the "1." prefix in the property name reflect the column to which the feature is applied. The F1 scores for all testing performed on all iterations of Stanford Classifier built-in feature sets:

Stanford Classifier F1 Scores for Binary Classification

Feature Set	POSITIVE	NEGATIVE
Baseline	0.737	0.759
chris1	0.802	0.790
chris2	**0.822**	**0.809**
chris4	**0.822**	**0.809**
dan1	0.730	0.743
dan2	0.808	0.793
jenny1	0.807	0.792

Despite the success of the classifier alone on a binary classification task, as the status update gets longer, misclassification seems to increase. In many cases, unigram features of a single article or preposition, have sentiments which are highly subjective based on their surrounding context. As hypothesized earlier, these longer status messages are in mostly sentence-style form. Thus, we expect that the POS tagger would be able to properly classify sentences such as these. We'll present examples of this in the next section.

Aware that an overabundance of positively or negatively labeled data in our training set could cause learning error, we did a quick analysis of the composition of both our training and test sets, as shown below.

Ratio of Labeled Classes in Training Data

Classes	POSITIVE	NEGATIVE
Count	1715	1785
Ratio	49.00%	51.00%

Ratio of Labeled Classes in Test Data

Classes	POSITIVE	NEGATIVE
Count	445	375
Ratio	54.27%	45.73%

CLASSIFIER PLUS POS TAGGER

The next set of features we turned to was POS tags. We used the Stanford POS tagger described above to provide POS tags for each word in the Facebook status message. The first set of features we considered were simply adding POS tags to each word in the sentence, increasing our F1 scores by ~5%. One benefit of this method is the disambiguation of certain words. For example, tagging “scratch” as either “scratch_VB” or “scratch_NN” resulted in different words for the classifier to consider. This allowed the classifier to weight words positively or negatively, depending on their part of speech.

Consider the following longer status update that the Classifier alone classified incorrectly, but was classified correctly when POS tags were included: *Click "LIKE" if I ever made you smile in your life. Set this as your status and see how many people you have made smile xox*

Classifier alone			Classifier with POS tags		
	NEGATIVE	POSITIVE		NEGATIVE	POSITIVE
…			…		
1-SW-in	-0.25	0.25	1-SW-this_DT	0.28	-0.28
1-SSHAPE-xxxxx	-0.28	0.28	1-SW-._.	0.26	-0.26
1-SW-this	0.34	-0.34	1-SSHAPE-xx_xXX	0.45	-0.45
1-SW-your	0.23	-0.23	1-SW-made_VBN	0.34	-0.34
1-SSHAPE-Xxx	0.35	-0.35	1-SSHAPE-XxX_xXX	-0.21	0.21
1-SW-Click	0.20	-0.20	1-SSHAPE-xxX_xX$	0.31	-0.31
1-SSHAPE-Xxxxx	-0.41	0.41	1-SSHAPE-._.	0.26	-0.26
1-SW-people	0.33	-0.33	1-SW-smile_VBP	-0.22	0.22
1-SW-you	-0.22	0.22	1-SSHAPE-X_XXX	0.28	-0.28
1-SSHAPE-X	0.34	-0.34	1-SW-you_PRP	-0.42	0.42
1-SW-smile	-0.72	0.72	1-SSHAPE-xxX_xXX	0.25	-0.25
…			1-SW-in_IN	-0.30	0.30
Prob:	0.66	0.34	1-SSHAPE-XXX_XX	-0.34	0.34
			1-SW-smile_NN	-0.22	0.22
			1-SW-have_VBP	-0.28	0.28
			1-SW-ever_RB	-0.32	0.32
			1-SW-many_JJ	0.24	-0.24
			1-SWP-your_PRP$-life_NN	-0.20	0.20
			…		
			Prob:	0.10	0.90

The Classifier alone heavily weighted words such as “this,” “Click,” “people,” and short capitalized words (such as “Set” and “Click”) as negative features, pushing the sentence slightly negative, despite its obvious positive sentiment. Once the words were POS tagged, however, the Classifier was able to determine the correct sentiment at a very strong level. The POS tags avoided the problem of particular words being weighted negative simply because they appeared in more negative samples.

Consider another, simpler example. The Classifier alone incorrectly classified *Cole made me breakfast* as a negative sentiment, while the Classifier plus POS tags classified *Cole_NNP made_VBD me_PRP breakfast_NN* correctly.

Classifier alone			Classifier with POS tags		
	NEGATIVE	POSITIVE		NEGATIVE	POSITIVE
CLASS	0.62	-0.62	CLASS	-0.71	0.71
1-SWP-made-me	0.02	-0.02	1-SWP-Cole_NNP-made_VBD	0.00	0.00
1-SW-me	-0.27	0.27	1-SW-me_PRP	-0.19	0.19
1-SWP-me-breakfast	0.00	0.00	1-SW-breakfast_NN	-0.15	0.15
1-SSHAPE-xx	-0.04	0.04	**1-SW-made_VBD**	**-0.06**	**0.06**
1-SW-breakfast	-0.09	0.09	1-SFW-Cole_NNP	0.00	0.00
1-SLW-breakfast	0.00	0.00	1-SSHAPE-xx_xXX	0.45	-0.45
1-SW-Cole	0.00	0.00	**1-SWP-made_VBD-me_PRP**	**0.03**	**-0.03**
1-SW-made	**0.10**	**-0.10**	1-SLW-breakfast_NN	0.00	0.00
1-SFW-Cole	0.00	0.00	1-SSHAPE-xxX_xXX	0.25	-0.25
1-SSHAPE-xxxxx	-0.28	0.28	1-SW-Cole_NNP	0.00	0.00
1-SSHAPE-xxxx	0.02	-0.02	1-SWP-me_PRP-breakfast_NN	0.00	0.00
1-SWP-Cole-made	0.00	0.00	1-SSHAPE-XxX_xXX	-0.21	0.21

1-SSHAPE-Xxxx	**0.18**	**-0.18**	1-SSHAPE-xxX_XX	0.15	-0.15
Total:	0.25	-0.25	Total:	-0.45	0.45
Prob:	0.62	0.38	Prob:	0.29	0.71

Here, we see that a word that had a relatively strong negative connotation, “made,” was given a positive connotation once it was assigned the VBD tag by the POS tagger. “Made me” retained its negative sentiment, likely due to the connotation of being forced to do something, but “made” itself become more positive. While “made” can have negative connotations, it can also have positive ones, such as when used in the context of creation instead of force. Also, the word shape “Xxxx” is deemed to be a negative feature. However, when the POS tags are added, there are more varieties of word shapes. The prevalence of Xxxx in negative status messages no longer incorrectly sways the status message to be classified as negative.

In an attempt to improve our classification even further, we next added a class of features that looked at the overall grammatical structure of the sentence. We considered the number of nouns (N), the number of verbs (V), the number of adjectives & adverbs (A), the ratio of nouns to adjectives, the ratio of verbs to adverbs, the number of terminals, and the number of exclamation points (E). The table below reflects a small sampling of possible combinations of these features that we tested. The best results occurred when we included features for the number of nouns, the number of verbs, the number of adjectives & adverbs, the ratio of nouns to adjectives, and the ratio of verbs to adverbs.

Trial	F1 Score: Negative	F1 Score: Positive
Classifier alone	0.809	0.822
+POS, no additional features	0.857	0.884
+POS with NVA	0.860	0.886
+POS with All	0.860	0.886
+POS with Ratios	0.857	0.884
+POS with Terminals & Ratios	0.855	0.882
+POS with NVAA	0.860	0.886
+POS with NVAE	0.856	0.882
+POS with NVA & Ratios	**0.863**	**0.888**
+POS with NVAE & Ratios	0.859	0.885

By adding the NVA and ratios of nouns to adjectives and verbs to adverbs features, we observed a ~1% increase in the F1 score. Consider the example *My work-husband's last day*. This is a sentence that a human would easily classify as negative: the speaker’s “work-husband” (a designation typically reserved for a close friend at work) is leaving.

Classifier + POS tags			Classifier + POS tags + NVA + ratios		
	NEGATIVE	POSITIVE		NEGATIVE	POSITIVE
CLASS	-0.71	0.71	3-Value (#A)	-0.02	0.02
1-SW-My_PRP$	0.09	-0.09	**5-Value (#V/#A)**	**0.05**	**-0.05**
1-SSHAPE-'xX_XX	-0.04	0.04	6-SWP-last_JJ-day_NN	0.22	-0.22
1-SLW-day_NN	0.04	-0.04	CLASS	-0.69	0.69
1-SSHAPE-XxX_X$	0.09	-0.09	6-SFW-My_PRP$	0.05	-0.05
1-SSHAPE-xx-_xXX	-0.06	0.06	1-Value (#N)	-0.04	0.04
1-SFW-My_PRP$	0.05	-0.05	6-SSHAPE-xx-_xXX	-0.02	0.02
1-SSHAPE-xx_xXX	0.45	-0.45	6-SW-last_JJ	0.05	-0.05
1-SW-day_NN	**-0.33**	**0.33**	**2-Value (#V)**	**0.09**	**-0.09**
1-SWP-'s_POS-last_JJ	0.05	-0.05	**4-Value (#N/#A)**	**0.01**	**-0.01**
1-SW-'s_POS	0.06	-0.06	6-SWP-'s_POS-last_JJ	0.05	-0.05
1-SW-last_JJ	0.05	-0.05	6-SW-My_PRP$	0.09	-0.09
1-SWP-last_JJ-day_NN	0.22	-0.22	6-SLW-day_NN	0.04	-0.04

Prob:	0.49	0.51	6-SSHAPE-xx_xXX	0.47	-0.47
			6-SSHAPE-'xX_XX	-0.05	0.05
			6-SW-day_NN	-0.30	0.30
			6-SW-'s_POS	0.09	-0.09
			6-SSHAPE-XxX_X$	0.09	-0.09
			Prob:	0.59	0.41

Here, we see the Classifier plus POS tags alone fails because of the strongly positive connotation of the word "day." However, the additional features added by the NVA and ratios were enough to pull the sentence from very slightly positive to the correct classification of negative. Here, the number of verbs (0.0), ratio of verbs to adverbs (0.0) and ratio of nouns to adjectives (2.0) suggested the sentence should be negative. The lack of descriptive words provided the Classifier with valuable insight into the sentiment of the sentence.

CLASSIFIER PLUS LDA

As stated, the motivation for incorporating a Gibbs LDA, as used by the Stanford Topic Modeling Toolkit, is to employ an unsupervised learning algorithm to discover latent features within our training data. Modifying a traditional LDA like Gibbs, such that the number of classes assigned to words within the status update is constrained to a fixed set of labels, is known as a Labeled LDA, and what we have referred to throughout this paper as simply "LDA." The output of the Stanford LDA maps a relative fractional count of the words in an update that were assigned to the POSITIVE label, and those mapped to the NEGATIVE label. For example, a training file with LDA output (`fbs.binomial.LDA.train`) might contain a line such as:

```
NEGATIVE        0.2     6.8     10.5 hours working at graduation = sun burnt +
sore throat + exhaustion = Party Pooper
```

With only two labels, POSITIVE and NEGATIVE, the LDA output in columns 2 and 3 above indicates that 0.2 words in the status update were assigned to the NEGATIVE label, and 6.8 were assigned to the POSITIVE label. These two real-valued columns became real valued features for our classifier to use. Combing the LDA and classifier resulted in the following feature set.

Stanford Classifier + LDA "Best" Features (Binary Classification)

Property	Value
1.realValued	true
1.binnedValues	-1.0, 0.0, 5.0, 10.0, 15.0, 100.0
1 .binnedValuesStatistics	true
2.realValued	true
2.binnedValues	-1.0, 0.0, 5.0, 10.0, 15.0, 100.0
2 .binnedValuesStatistics	true
3.useClassFeature	true
3.splitWordsRegexp	[,]
3.useSplitWords	true
3.useSplitWordPairs	true
3.useSplitFirstLastWords	true
3.useSplitPrefixSuffixNGrams	true
3.splitWordShape	chris4

For columns 1 and 2 above, the "binnedValues" feature of the Stanford Classifier prevents data scarcity by discretizing the LDA output into ranged bins. "binnedValuesStatistics" tells the Classifier to log data regarding the collection sizes of the bins for our later analysis.

Like the Classifier, the LDA also has customizable features. Unfortunately, LDA did not perform to our expectations. The default LDA features often rendered many status updates unreadable by the LDA. Of the 3,500 status updates used in the training set, the LDA returned metrics for only

1,446 of them, less than 40% of the total. In attempting to give the LDA more data to work with, we implemented the "relaxed LDA feature set" below, assuming the elimination of *DocumentMinimumLengthFilter* would lead to a wealth of usable data. However, elimination of this feature in particular resulted in the LDA throwing a runtime exception.

Default LDA features	Relaxed LDA feature set

// indicates status update is in col 2
Column(2)
// lowercase everything
CaseFolder
// tokenize on spaces characters
SimpleEnglishTokenizer
// ignore non-words and non-numbers
WordsAndNumbersOnlyFilter
// collect counts (needed below)
TermCounter
// take terms with >=3 characters
TermMinimumLengthFilter(3)
// filter terms in <4 docs
TermMinimumDocumentCountFilter(4)
// filter out 30 most common terms
TermDynamicStopListFilter(30)
// take only docs with >=5 terms
DocumentMinimumLengthFilter(5)
	// indicates status update is in col 2
	Column(2)
	// tokenize on spaces characters
	SimpleEnglishTokenizer
	// collect counts (needed below)
	TermCounter
	// take terms with >=3 characters
	TermMinimumLengthFilter(3)
	// filter terms in <4 docs
	TermMinimumDocumentCountFilter(4)
	// take only docs with >=5 terms
	DocumentMinimumLengthFilter(5)

The distribution of bins of LDA data shows heavily skewed 0-counts, as shown below:

```
BinnedValuesStatistics for column 1
Val-(-1.0,0.0] = {POSITIVE=1109.0, NEGATIVE=1068.0}
Val-(0.0,5.0] = {POSITIVE=564.0, NEGATIVE=293.0}
Val-(10.0,15.0] = {POSITIVE=4.0, NEGATIVE=82.0}
Val-(15.0,100.0] = {POSITIVE=1.0, NEGATIVE=24.0}
Val-(5.0,10.0] = {POSITIVE=37.0, NEGATIVE=318.0}

BinnedValuesStatistics for column 2
Val-(-1.0,0.0] = {POSITIVE=994.0, NEGATIVE=1180.0}
Val-(0.0,5.0] = {POSITIVE=262.0, NEGATIVE=532.0}
Val-(10.0,15.0] = {POSITIVE=101.0, NEGATIVE=8.0}
Val-(15.0,100.0] = {POSITIVE=40.0, NEGATIVE=1.0}
Val-(5.0,10.0] = {POSITIVE=318.0, NEGATIVE=64.0}
```

Observe from the example above that LDA only recognized 7 words despite there being at least 11 legitimate words. In an attempt to improve the word count parsed by the LDA, we wanted to implement greater flexibility in word tokenization than the simple "space" used by the *SimpleEnglishTokenizer*. After consulting with Evan Rosen, one of the Stanford Topic Modeling Toolbox developers, he acknowledged that the LDA is not able to handle some input characters, only implements a couple of the scalanlp.text.tokenize methods, and remains a primarily beta effort.

The final result of LDA classification of binary sentiment for Facebook status updates is disappointing. Although we ran a number of feature and iteration combinations, none of them showed an improvement, even over the MaxEnt classifier alone. Since Classifier feature sets "chris2" and "chris4" had performed equally before, we ran both sets with the LDA to determine differentiation, if any. The final results, highlighting our best overall binary classification model overall, are below.

Trial	F1 Score:	F1 Score:

	Negative	Positive
Classifier alone	0.809	0.822
+LDA (defaults, chris2)	*0.670*	*0.667*
+LDA (defaults, chris4)	*0.670*	*0.667*
+LDA (defaults, binned, chris4)	0.665	0.659
+LDA (defaults, 5k, chris2)	0.669	0.667
+LDA (defaults, 5k, chris4)	0.669	0.667
+LDA (relaxed, chris2)	0.637	0.600
+ LDA (relaxed, chris4)	0.638	0.602
+ LDA (relaxed, 5k, chris2)	0.630	0.587
+ LDA (relaxed, 5k, chris4)	0.630	0.587
+POS with NVA & Ratios	**0.863**	**0.888**
+POS with NVA & Ratios + LDA	0.753	0.773

Due to the errors we encountered with the LDA itself, we're unable to make any assertive conclusions. Is the serious degradation of scores simply implementation error, or is the use of an LDA ultimately of no benefit in the context of our sentiment analysis topic?

FOUR CLASSES: Happy, Unhappy, Skeptical, Playful

While binary classification can be helpful when your objective is general trending analysis, deeper semantic extraction requires more granular sentiment analysis. To say that a person feels more positive about some thing than negative conveys little understanding of the nature of the positivity. Is there happiness, or merely tolerance? Is there playfulness, or merely acknowledgment? For this project, we chose to replace our binary classification task with a four-class classification of happy, unhappy, skeptical, and playful sentiments. As discussed previously, these were primarily chosen based on the amount of data in those emoticon categories to generate sufficient training and test set sizes. As before, each of the four classes was truncated to provide a roughly even distribution of each across training and test sets.

Having experimentally derived a "best model" on the binary classification task, our objective in this section focused on determining whether that model scales to the harder task of multi-label classification. For good measure, we did run LDA again because we were curious to see if multiple classes would showcase its modeling strengths, perhaps indicating LDA was inherently weaker with fewer class labels to which it could assign words.

CLASSIFIER

Using the same set of "best features" as with the binary classification task, our baseline F1 scores using only the MaxEnt classifer are below. Although "chris4" showed a 0.017 net degradation versus "chris2", we decided to stick with "chris4" throughout the remainder of our testing.

Stanford Classifier F1 Scores for Multi-Label Classification

Feature Set	HAPPY	PLAYFUL	SKEPTICAL	UNHAPPY
binary baseline	0.737 (POSITIVE)	0.759 (NEGATIVE)	➢ *For prior reference*	
binary best	0.888 (POSITIVE)	0.863 (NEGATIVE)		
chris2	0.480	0.496	0.736	0.667
chris4	**0.476**	**0.496**	**0.731**	**0.659**

Consistent with our expectation, the multi-label classification task proved more difficult than the simpler binary classification task.

Ratio of Labeled Classes in Training Data

Classes	HAPPY	PLAYFUL	SKEPTICAL	UNHAPPY
Count	843	831	813	813
Ratio	25.54%	25.18%	24.63%	24.63%

Ratio of Labeled Classes in Test Data

Classes	**HAPPY**	**PLAYFUL**	**SKEPTICAL**	**UNHAPPY**
Count	60	72	90	90
Ratio	19.23%	23.08%	28.85%	28.85%

Although the test data contains slightly higher proportions of "skeptical" and "unhappy" data, the training set maintains a relatively even distribution such that we aren't concerned about training bias toward that data.

CLASSIFIER PLUS LDA

Having been previously disappointed with LDA results during binary classification, we were optimistic that perhaps the inclusion of more labels to classify would give the LDA some sense of purpose. We ran tests comparing the same default LDA features and relaxed LDA features as described in the binary classification section, as well the default of 1,500 iterations versus 5,000 iterations. In addition, we decided to perform an intermediate experiment to determine whether LDA results would improve if augmented with POS tags and only LDA features (e.g. no POS features). The net result is somewhere in-between, better than the classifier-only, but far worse than the combination of POS and classifier. The results of these tests, along with an LDA-only baseline matching the features from the binary LDA tests, are summarized below. In the sections which follow, LDA discussions are assumed to include POS-tagged data.

Trial	F1 Score: Happy	F1 Score: Playful	F1 Score: Skeptical	F1 Score: Unhappy
Classifier-only	**0.476**	**0.496**	**0.731**	**0.659**
+LDA (no POS, best)	0.392	0.500	0.637	0.536
+LDA (relaxed, chris2)	0.451	0.614	0.847	0.547
+LDA (relaxed, chris4)	0.451	0.614	0.847	0.547
+LDA (5k,relax,chris2)	0.454	0.650	0.857	0.554
+LDA (5k,relax,chris4)	0.451	0.644	0.857	0.560
+LDA (default, chris2)	0.453	0.655	0.880	0.565
+LDA (default, chris4)	0.453	0.655	0.876	0.557
+LDA (binned, chris4)	0.416	0.605	0.859	0.485
+LDA (default,5k,chris2)	**0.471**	**0.650**	**0.886**	**0.559**
+LDA (default,5k,chris4)	**0.471**	**0.650**	**0.886**	**0.559**

Interestingly, the probability mass shifts from "happy" and "unhappy" to "playful" and "skeptical" when using LDA and POS tags (no POS features). The overall improvement is still not what we had hoped for, but does reflect an 8% improvement versus the classifier-only. While we'd like to give LDA credit for doing its job, we suspect the increase is more attributable to the POS tags rather than the number of labels. Nonetheless, considering that LDA alone generally degrades classifier-only results, the improvement with POS tags is noteworthy.

A comparison of LDA versus classifier-only results also reveals the reason for higher "skeptical" and "unhappy" scores: the distinction between playful and happy is subtle, and both have low confidence when distinguishing between the two. One might argue the same relationship between skeptical and unhappy, but the semantics are more polarized in the latter case. In the former, there is arguably a much higher correlation between playfulness and a state of happiness. Consider the following "playful" example: *lol, you gotta see this! w=HtTp://u.MavRev.cO09dp2*. The classifier gives this status update a "skeptical" label, which is clearly wrong. The LDA, on the other hand, gives it a "happy" label. This is still wrong, but at least LDA gets closer to the underlying sentiment expressed by the update.

LDA Classification example: *lol, you gotta see this! w=HtTp://u.MavRev.cO09dp2*

Classifier-only

```
                     HAPPY PLAYF SKEPT UNHAP
…
1-SW-this!           -0.46  0.16  0.43 -0.12
1-SWP-see-this!      -0.24  0.01  0.35 -0.13
1-SWP-you-gotta      -0.16 -0.11  0.33 -0.06
1-SW-you              0.00  0.53  0.23 -0.75
1-SWP-gotta-see      -0.18 -0.06  0.31 -0.07
1-SWP--you            0.01 -0.32  0.33 -0.01
1-SW-lol             -0.21  0.23  0.08 -0.10
…
Prob:                 0.22  0.18  0.55  0.05
Sentence Prob for SKEPT: 0.5473878585366191
```

Classifier with LDA

```
                     HAPPY PLAYF SKEPT UNHAP
…
1-Value               0.79  0.14  0.02 -0.25
2-Value              -0.10  0.95  0.37 -0.27
3-Value              -0.01  0.63  0.88 -0.51
4-Value               0.16  0.03  0.11  0.81
5-SW-you_PRP          0.03  0.39 -0.10 -0.32
5-SSHAPE-xx_xXX       0.31 -1.77  0.01  1.37
5-SW-this_DT         -0.37  0.52  0.18 -0.33
5-SW-lol_NN           0.01  0.22 -0.03 -0.21
5-SW-!_.              0.41  0.29 -0.21 -0.50
5-SSHAPE-!_.          0.41  0.29 -0.21 -0.50
5-SSHAPE-=_XXX        0.23 -0.21  0.39 -0.41
5-SSHAPE-x_XX         0.04 -0.73  0.40  0.29
…
Prob:                 0.42  0.39  0.17  0.01
Sentence Prob for HAPPY: 0.6659759502662165
```

In the classifier-only classification, the features which include "this!" seem strangely mapped to "skeptical". Looking at other status updates with "this!", it turns out we encounter other examples, such as "Guys! You have to see this! Disney's most shocking hidden message!...", where the containing sentence *is* labeled as "skeptical." "You gotta see" might understandably be associated with skepticism, and "lol" is the only feature we would expect to associate with "playful" that actually is. At least we take comfort that the classifier is not very confident of its labeling, with a likelihood of only 54.7%

Looking at the LDA feature weights, the "1-Value", "2-Value", "3-Value", and "4-Value" feature names reflect the LDA word-label ratio counts, mapping to class labels PLAYFUL, SKEPTICAL, UNHAPPY, and HAPPY, respectively. Tellingly, "playful" (1-Value) is strongly associated with "happy", with "playful" only a distant second association. Similarly, "unhappy" (3-Value) is most closely associated with "skeptical" but, strangely, also surprisingly correlated with "playful". Stranger still, "skeptical" (2-Value) is also most strongly correlated to "playful", with its true association to itself a distant second. And "happy" (4-Value) is most strongly correlated with "unhappy," with itself second.

The classifier struggles against this weirdness but is helped by the POS tagger. "this" is separated from "!" by the tagger, resulting in a significant shift of probability mass from "skeptical" to "happy" and "playful". Classifier word shape features seem to interpret POS tags themselves as being closely correlated to "skeptical," and "playful" tagging is heavily penalized by the presence of xx_xXX word shapes, although it's not apparent what word that corresponds to. Most problematic, however, is that "!" is more strongly associated with "happy" than with "playful". The "!" can be explained away as a likely symptom of the training samples containing an unexpected prevalence of "!" in "skeptical" labels. LDA must be given credit for a being a factor for shifting probability mass toward "happy" and "playful", as described in the preceding paragraph, and the POS tagger for allowing us to learn the difference between "!" and "this!". We can also take comfort in the fact that the classifier at least acknowledged its uncertainty, scoring the labeling with 66.5% likelihood In hindsight, the same benefit induced by POS tagging could have been realized by the classifier alone with letter shape features around sentence and word endings.

CLASSIFIER PLUS POS TAGGER

The process for adding POS tags was exactly the same as in the binary case above, but this time we observed F1 score improvements ranging from 2% to 20%, depending on the class.

Consider the following "unhappy" sentence that the Classifier alone classified incorrectly as "happy," but was classified correctly when POS tags were included: *lost photos from Paris stupid replace command*.

Classifier alone					Classifier with POS tags				
	HAPPY	PLAYF	SKEPT	UNHAP		HAPPY	PLAYF	SKEPT	UNHAP
CLASS	-0.50	0.90	1.05	-0.91	CLASS	0.43	1.48	-1.60	-0.49
1-SW-stupid	0.23	-0.19	-0.11	0.06	1-SW-from_IN	0.02	-0.04	-0.21	0.23
1-SW-from	-0.19	0.01	0.24	-0.05	**1-SSHAPE-xx_xXX**	**0.38**	**-1.69**	**0.10**	**1.37**
1-SW-lost	-0.42	-0.23	-0.25	0.90	1-SW-photos_NNS	0.13	0.06	0.04	-0.23
1-SW-photos	-0.01	0.15	0.26	-0.40	1-SW-lost_VBN	0.02	-0.02	-0.10	0.11
1-SW-Paris	-0.11	-0.05	0.03	0.13	1-SSHAPE-xxX_xXX	-0.03	-0.35	-0.39	0.76
1-SWP-photos-from	0.15	-0.16	0.08	-0.07	1-SSHAPE-XxX_xXX	0.49	-0.40	-0.06	0.09
1-SWP-Paris-	-0.10	-0.01	-0.02	0.14	1-SW-stupid_JJ	0.23	-0.14	-0.06	-0.03
1-SW-replace	-0.06	-0.00	-0.00	0.06	1-SWP-photos_NNS-from_IN				
1-SSHAPE-xxxxx	**0.84**	**-1.16**	**-0.83**	**1.03**		0.17	-0.12	0.05	-0.10
1-SWP--stupid	0.21	-0.09	-0.03	-0.09	1-SW-Paris_NNP	-0.10	-0.03	-0.01	0.14
1-SSHAPE-xxxx	0.07	-0.29	-0.32	0.56	Prob:	0.46	0.02	0.01	0.51
1-SSHAPE-Xxxxx	**1.01**	**-0.60**	**-0.10**	**-0.16**					
1-SW-	0.10	-0.09	-0.05	0.06					
1-SSHAPE-	0.10	-0.09	-0.05	0.06					
Prob:	0.44	0.02	0.10	0.44					

Here, the word shape features without the POS tags causes the classifier to classify the sentence as happy, despite its obvious unhappy sentiment. The addition of the POS tags removes those errant features and replaces them with other features that are slightly more rare and so better equipped to determine sentiment.

As described above, we next added the same set of POS tag class of features that looked at the overall grammatical structure of the sentence. The table below reflects a small sampling of possible combinations of these features that we tested. The best results occurred when we included features for the number of nouns, the number of verbs, the number of adjectives & adverbs, and the number of exclamation marks.

Trial	F1 Score: Happy	F1 Score: Playful	F1 Score: Skeptical	F1 Score: Unhappy
Classifier alone	0.476	0.496	0.731	0.659
+POS, no add'l features	0.503	0.661	0.954	0.656
+POS with NVA	0.528	0.656	0.960	0.681
+POS with All	0.514	0.624	0.954	0.674
+POS with Ratios	0.500	0.645	0.876	0.543
+POS with Terminals & Ratios	0.510	0.619	0.949	0.652
+POS with NVAA	0.524	0.640	0.960	0.674
+POS with NVAE	**0.521**	**0.677**	**0.960**	**0.685**
+POS with NVA & Ratios	0.503	0.624	0.960	0.670
+POS with NVAE & Ratios	0.503	0.624	0.960	0.670

By adding the NVAE features, we observed a 1% – 3% increase in the F1 score. Consider the example *thinks he has the flu!*. This is a sentence that a human would easily classify as unhappy: no one wants the flu, but could easily confuse a classifier given it appears "happy" on paper.

```
Classifier + POS tags
                  HAPPY PLAYF SKEPT UNHAP
CLASS              0.43  1.48 -1.60 -0.49
1-SW-flu_NN       -0.19 -0.24 -0.20  0.64
1-SLW-!_.          1.21 -0.48 -0.79  0.07
1-SWP-he_PRP-has_VBZ
                   0.25 -0.04 -0.04 -0.16
1-SFW-thinks_VBZ
                  -0.08  0.24 -0.04 -0.12
1-SW-the_DT        0.38 -0.11 -0.26 -0.01
1-SWP-has_VBZ-the_DT
                   0.17 -0.10 -0.05 -0.03
1-SSHAPE-xx_xXX 0.38 -1.69  0.10  1.37
1-SW-!_.           0.37  0.31 -0.18 -0.47
1-SWP-flu_NN-!_.
                  -0.05 -0.00 -0.00  0.06
1-SW-thinks_VBZ-0.16  0.54 -0.10 -0.28
1-SSHAPE-xxX_xXX
                  -0.03 -0.35 -0.39  0.76
1-SW-he_PRP       -0.04 -0.18  0.06  0.16
1-SW-has_VBZ      -0.78  0.06  0.05  0.67
1-SSHAPE-xxX_XX
                  -0.12 -0.05  0.05  0.12
1-SSHAPE-!_.       0.37  0.31 -0.18 -0.47
```

```
Classifier + POS tags + NVAE
                  HAPPY PLAYF SKEPT UNHAP
3-Value (#A)       0.04  0.29  0.15 -0.03
5-SWP-he_PRP-has_VBZ
                   0.21 -0.04 -0.04 -0.13
5-SW-!_.           0.37  0.20 -0.20 -0.46
5-SSHAPE-!_.       0.37  0.20 -0.20 -0.46
CLASS              0.46  1.45 -1.56 -0.36
5-SW-he_PRP        0.01 -0.20  0.09  0.11
5-SW-has_VBZ      -0.70  0.03  0.12  0.56
5-SWP-flu_NN-!_.
                  -0.06 -0.00 -0.00  0.06
5-SSHAPE-xx_xXX 0.32 -1.74  0.07  1.44
5-SSHAPE-xxX_XX
                  -0.06 -0.05  0.09  0.03
1-Value (#N)       1.04  1.05  0.96  0.84
5-SWP-has_VBZ-the_DT
                   0.14 -0.10 -0.06  0.01
5-SFW-thinks_VBZ
                  -0.06  0.23 -0.03 -0.14
5-SW-flu_NN       -0.22 -0.23 -0.20  0.64
4-Value (#E)      -0.29 -0.12 -0.21 -0.27
2-Value (#V)       0.03  0.24  0.10  0.53
5-SW-the_DT        0.41 -0.18 -0.23 -0.00
5-SSHAPE-xxX_xXX
                   0.07 -0.37 -0.30  0.60
5-SLW-!_.          1.26 -0.55 -0.80  0.11
5-SW-thinks_VBZ
                  -0.13  0.54 -0.09 -0.32
```

Here, we see the Classifier plus POS tags alone fail because of the strongly positive connotation of the exclamation point, selecting happy with a probability of 0.542. However, the additional features added by the NVAE features are enough to pull the sentence from very slightly happy to the correct classification of unhappy, selecting the proper sentiment with a probability of 0.500. Here, the number of nouns (1.0), and the number of verbs (2.0) suggested the sentence should be unhappy.

Error Analysis

"Whenever I go on a ride, I'm always thinking of what's wrong with the thing and how it can be improved." –Walt Disney

DATA COLLECTION

Ideally, we wanted to collection 1,000,000 raw status updates. We knew going in that we would label via emoticon, and assumed a signal-to-noise ratio of 90%. We anticipated duplicates as a result of shared friends by using the sort –u(nique) Linux command. However, we neglected to consider that many Facebook users re-post their status updates with only slight changes, or that some status updates are common among many different users. For example, one user posted a succession of the following status updates:

2 days
2 days !
2 days!!!!
2 days baby!!

On major holidays, many similar status messages can be found, such as these from many different users:

> *happy mother's day to all the mothers out there*
> *Happy Mother's Day to all the moms out there...*
> *Happy Mother's Day to all my mommy friends out there!*
> *Happy Mothers Day*
> *Happy mother's day to all the mommies out there*

Contrast the above with a sentence like “I’m not happy!”: would this be classified correctly? We chose to retain the distinction of capitalization, as well as punctuation, but this could also hurt us if frequent repetition in a certain sentiment inappropriately influences our feature weights. In this specific example, it turns out that our model does appropriately label “I’m not happy!” as NEGATIVE, with 60.2% confidence. Understandably, “not” is heavily correlated with “negative”, assigned a weight of 1.26. Another example, “2 days until I die!”, does not fare so well. This time, the classifier declares that sentence to be “positive,” with 91.2% confidence. Surprisingly, it’s not because of the word “die,” which is assigned no weight. Instead, “2 days” associates strongly with “positive,” as do words ending with “!”. Would we have classified this correctly had we implemented better filtering among updates with common word prefixes and sentiments? We propose a solution to this in the “Pre-processing” section of *Conclusions and Future Work*.

PREPROCESSING

Although we had a decent-sized raw corpus, we had no time to hand label or verify that all of the data were properly labeled by emoticons. We spot checked a few samples for sanity and were generally pleased with the results. Still, the fact that emoticons served as our sentiment barometer also eliminated what could have been an important learned feature. Imagine a word- or letter-shape feature that models emoticons, or even features for specific emoticons. Lack of time to hand-label our data probably negatively impacted our results. Further, the need to remove the emoticons once labeled led to complicated regular expression filtering that was not foolproof. In some cases, we did observe emoticons that snuck their way into sentences. In particular, we observed a handful of sentences labeled as NEGATIVE that had had :(removed, but not :). Sufficient time to hand-label would have eliminated this error altogether.

Additionally, we assumed garbage strings such as URLs would be a source of error. For example, many status updates we collected had URLs in them. In turns out that sparsity actually helps here. Since nearly every link is unique, feature weights turn out to be 0. Similarly, we tokenized only on space and comma characters when, perhaps, we should have tokenized on all characters which provide no sentiment insight. We also focused on word shape features and did not include any letter shape features. Such a feature may have given more weight to borderline cases. Letter shape features may also have been able to model word-based prefixes and suffixes.

BINARY

Despite the successes in the binomial case, the Classifier + POS tags + other POS features still misclassified many sentences. Consider the example *internet working again*.

	NEGATIVE	POSITIVE
3-Value	-0.02	0.02
5-Value	0.05	-0.05
6-SLW-again_RB	0.24	-0.24
CLASS	-0.69	0.69
1-Value	-0.04	0.04
6-SW-working_VBG	**0.64**	**-0.64**
6-SW-again_RB	0.21	-0.21
6-SW-internet_NN	0.15	-0.15

```
2-Value                        0.09    -0.09
4-Value                        0.01    -0.01
6-SSHAPE-xxX_xXX               0.19    -0.19
6-SSHAPE-xx_xXX                0.47    -0.47
Prob:                          0.93     0.07

internet_NN working_VBG again_RB  POSITIVE      NEGATIVE      0.9307304154458833
```

The predominant negative feature here is still the word "working." Adding POS tags and features do not teach the Classifier that this sentence is positive. While working is generally considered negative, in the case of the Internet, most users prefer it to be working! Context, such as knowing working relates to the Internet as opposed to a job, may help improve examples such as this.

We considered using the Stanford Parser to incorporate additional grammatical features, but found by spot checking examples that the Stanford Parser, trained on PTB data, did not do a great job of determining dependencies for Facebook status messages. This makes sense, considering the type of language used in the WSJ is very different from the type of language used on Facebook. To correct this issue, a new training corpus would need to be created by parsing Facebook status messages.

MULTICLASS
The multiclass case also has room for improvement. Consider the example *looks like im waking up early next semester*.

```
                          HAPPY     PLAYFUL   SKEPTICAL UNHAPPY
3-Value                    0.04      0.29      0.15     -0.03
CLASS                      0.46      1.45     -1.56     -0.36
5-SW-early_RB             -0.12     -0.19     -0.19      0.50
5-SSHAPE-xx_xXX            0.32     -1.74      0.07      1.44
5-SW-next_JJ              -0.19      0.17      0.37     -0.35
5-SSHAPE-xx_XX            -0.03     -0.07     -0.22      0.29
5-SWP-looks_VBZ-like_IN   -0.06      0.03     -0.01      0.04
2-Value                    0.03      0.24      0.10      0.53
5-SSHAPE-xxX_xXX           0.07     -0.37     -0.30      0.60
5-SW-semester_NN           0.07     -0.08     -0.05      0.07
5-SW-im_NN                -0.36     -0.03      0.41     -0.02
1-Value                    1.04      1.05      0.96      0.84
5-SW-up_RB                -0.04     -0.02     -0.14      0.19
5-SW-looks_VBZ             0.15     -0.01     -0.06     -0.08
4-Value                   -0.29     -0.12     -0.21     -0.27
5-SW-waking_VBG           -0.07     -0.03     -0.01      0.11
5-SWP-up_RB-early_RB      -0.01     -0.01     -0.01      0.04
5-SW-like_IN              -0.24     -0.03     -0.12      0.39
Prob:                      0.04      0.03      0.01      0.92

looks_VBZ like_IN im_NN waking_VBG up_RB early_RB next_JJ semester_NN
       SKEPTICAL     UNHAPPY       0.913034316024457
```

Here, the number of nouns in the status message (2.0), and the words "next" and "im," pull the sentence toward unhappy, even though the author used a skeptical (:-/) smiley. There is a very fine line here between whether this is truly a skeptical message or an unhappy message. Even a human would struggle to classify this status message.

Conclusions and Future Work

If the long-term vision of this project is sentiment classification of Facebook status updates, with a correlation to topic, we've barely scratched the surface of work to be done in pursuit of such a goal. Within the broader vision, we constrained the scope of this project to evaluation of how well the "best" binary classification model would scale to the harder problem of multi-label classification. Would LDA be the key, would POS, would feature sets, or would it be some combination of all of these? We've shown that, in fact, our best model does scale to the mutli-class task in the sense that a similar set of features proved to result in the best classification results for both problems. Facebook status updates, more generously capped at 420 characters, benefited tremendously from POS tagging given their more sentence-like structure. Surprisingly, LDA only hurt our results but we acknowledge that, due to inherent problems with the LDA code itself, these results are ultimately inconclusive.

Of course, with an increased number of labels comes increased uncertainty. While the models scaled in terms of relevant feature sets, misclassification rose, as expected. To mitigate this, we note the following areas in which our work could be extended and improved:

- Semantic Analysis: The next major phase of this project would be to classify the topic of status updates. Correlating sentiment and topic, particularly in the use of a parser to determine noun or verb of interest, would allow for true semantic analysis. We believe this has wide-ranging use for marketing/adware applications and sociological data mining for policy makers and world organizations
- Data Collection: As mentioned in error analysis, hand-labeling our Facebook corpus rather than using emoticons as proxy for sentiment would allow us to use emoticons as features instead. This would remove a source of potential error (multiple smileys or "fake smileys") and replace it with what would likely be a very strong set of features. Hand-labeling would also have allowed us to use much more of the over 62,000 status updates we collected, resulting in more reliable model performance data
- Labeling: Another angle to consider is our very approach to labeling. For example, our multi-class labeling actually resulted in 4,382 "happy" sentences, 1,284 "playful" sentences, 1,254 "skeptical" sentences, and 903 "unhappy" sentences. To prevent learning from being unduly influenced by one sentiment over another, we capped each of these at 903 in the training set. But was that the correct approach? Of the total usable samples (e.g. ones we could auto-label using emoticons), is it not valuable to model the natural distribution of sentiments? Is it the case that people are more likely to update their Facebook status when they are happy about something than when they are unhappy? Are they more likely to distance themselves from Facebook when unhappy? With more time, it would have been an interesting experiment to use this distribution for the classifier, LDA, or both. It could have had a significant impact, especially in low-confidence examples. We could also have employed Laplace or other technique to smooth this distribution, creating the possibility for "unseen" label class.
- Pre-processing: Implementing smarter regular expression filtering to weed out non-alphanumeric characters that do not convey sentiment would increase the amount of relevant data and possibly improve our results. Also, providing a more advanced method for removing duplicates, such as using an alignment model to remove the two or three highest scored "translations" of the current status update avoids the problem of overweighting certain words/features.
- Classifier: We explored only the word feature sets provided as part of the Stanford Classifier package. But the source is all there to add a custom feature set. It's possible our feature wish-list could have been implemented via character-level shapes instead of creating new features. In our analysis of LDA performance in the multi-label classification task, we observed LDA correlations that model inherent ambiguities in the labels, such as between "happy" and "playful." One idea is to create a feature that represents the joint over two or more LDA features. In the LDA case, the misclassification may have been reduced had we been able to consider "happy" and "playful" together, or even "happy" and "unhappy," in the hopes of polarizing the feature weight assigned to these two.

- LDA: Clearly, the LDA results observed here bring down the overall score of our model substantially. As described, one restrictive feature could not be removed due to an exception thrown. The LDA does not implement RegExp tokenizers. Also, the data sparsity problem results in highly skewed label associations, increasing misclassification. Fixing these issues would hopefully result in useful features from the LDA, improving our F1 scores.
- Feature Engineering: Most of the features used in our "best" model may be considered unigram or, at the most, bigram in nature. In addition to the joint over LDA result columns proposed above, it might be similarly useful to compute the joint over both POS features and LDA features. For example, are HAPPY updates more likely to have a higher count of adverbs? Are SKEPTICAL updates more likely to have higher word and nouns counts? The Stanford Classifier grouping features may have been able to model this, but we did not have time to explore this capability.
- Parser: Training on Facebook data instead of PTB data would allow the Stanford Parser to provide accurate dependency data and open up more options for possible features. The language used in the WSJ is so different from the language used on Facebook, that the dependencies from the Stanford Parser were not accurate enough to justify using.

References

Dwergs. *What's the maximum length of a Facebook status update?* February 19, 2009. http://reface.me/status-updates/whats-the-maximum-length-of-a-facebook-status-update/

Eldon, Eric. *ComScore, Quantcast, Compete, Nielsen Show a Strong December for Facebook Traffic in the US*. January 22, 2010. http://www.insidefacebook.com/2010/01/22/comscore-quantcast-compete-show-a-strong-december-for-facebook-traffic-in-the-us/

Go, Alec; Huang, Lei; Bhayani, Richa. *Twitter Sentiment Analysis*. Stanford University, Spring 2009, CS224N Natural Language Processing, Professor Chris Manning. http://nlp.stanford.edu/courses/cs224n/2009/fp/3.pdf

Harris, Jonathan; Kamvar, Sep. *We Feel Fine*. August 2005. http://www.wefeelfine.org

O'Connor, Brendan. Balasubramanyan, Ramnath; Routledge, Bryan R.; Smith, Noah A. *From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series*. Carnegie Mellon University, 2010. http://anyall.org/oconnor_balasubramanyan_routledge_smith.icwsm2010.tweets_to_polls.pdf

Pang, Bo; Lee, Lillian. *Opinion Mining and Sentiment Analysis*. Cornell University, 2008. http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf

Ramage, Daniel; Dumais, Susan; Liebling, Dan. Characterizing Microblogs with Topic Models. Stanford University. 2010. http://nlp.stanford.edu/pubs/twitter-icwsm10.pdf

Ramage, Daniel; Hall, David; Nallapati, Ramesh; Manning, Christopher D. *Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora*. Stanford University, 2009. http://www.stanford.edu/~dramage/papers/llda-emnlp09.pdf