Vector-based Sentiment Analysis of Movie Reviews

Ian Roberts and Lisa Yan
{iroberts, yanlisa}@stanford.edu

ABSTRACT

Sentiment analysis is an important step towards comprehension in natural language processing. Movie reviews are a convenient source of highly polarized sentences for use in sentiment analysis.

Achieving a high level of accuracy in the sign of non-neutral sentiment is a challenge. When the problem is expanded to choosing one of 5 sentiment levels the problem becomes significantly harder. Among the models that we tested, softmax regression with a bag of phrases feature set provided the best 5-bin error rate, while SVM with the same bag of phrases features and a simple word sentiment sum model provided the best error rate on sign prediction.

1. INTRODUCTION

We investigate sentence sentiment using the Pang and Lee dataset as annotated by Socher, et al. [1]. Sentiment analysis research focuses on understanding the positive or negative tone of a sentence based on sentence syntax, structure, and content. Previous research used a tree-based model to label sentence sentiment on a scale of 5 points. Our project takes a different approach of abstracting the sentence as a vector and apply vector classification schemes. We explore two components: first, we would like to analyze the use of different sentence representations, such as bag of words, word sentiment location, negation, etc., and abstract them into a set of features. Second, we would like to classify sentence sentiment using this set of features and compare the effectiveness of different models. While sentiment polarity was analyzed in a previous year's project, we would like to explore 5 degrees of sentiment labeling (Figure 1).

We chose to investigate sentiment of movie reviews which could be compared to numeric movie ratings. We looked at a variety of models and feature types to attempt to capture the context important for accurate sentiment decoding of the phrases of a sentence. While a tree-based feature set by Socher et al. [1] exists, we also wanted to explore how linear feature vectors would fare for sentence sentiment classification .

2. DATASET

We used the datasets from Stanford's Sentiment Analysis website which split the sentences into a training set and test set and provided a dictionary of sub-phrases and unique words (Table 1). All phrases (and words) have a sentiment label as well, which was determined in the original dataset via Amazon Mechanical Turk. Sentiment is labeled on a 5 point scale of 1 (most negative) to 5 (most positive), with 3 being neutral; a distribution of the sentence sentiment labels is in Figure 2.

5 (most positive)	*It was a breathtaking movie.*
4 (positive)	*It was a good movie.*
3 (neutral)	*It was a movie.*
2 (negative)	*It was a bad movie.*
1 (most negative)	*It was a terrible movie.*

Figure 1: Rating sentence sentiment on a 5-point scale.

	Training Set	Test Set
Sentences	8544	2210
	Dictionary	
	Phrases	Unique Words
Elements	239232	22348

Table 1: Stanford Sentiment Analysis dataset.

The dataset also includes a sentiment labeled partitioning of each sentence on a parse tree. Recursive node recombination can therefore be trained at all levels rather than just with the root sentiment.

3. APPROACH

3.1 Features

In Figure 3 we examine the distribution of individual word sentiments within sentences with each level of overall sentiment. The distributions are nearly identical with highly positive words showing up in highly negative sentences just as often as in highly positive ones. This provides a clear indication that phrases and syntax are key to sentence sentiment analysis. While the sentiments of individual words and phrases were labeled in our dataset the lack of correlation between word sentiment and sentence sentiment led us to ignore this labeling for some of our features.

We now overview our vector-based features, where we represent the i^{th} sentence in our dataset as a vector $x^{(i)}$ over

Feature Name	Domain of $x^{(i)}$	$x_j^{(i)}$
Bag of Words	$\{0,1\}^{\lvert V_W \rvert}$	$\mathbf{1}\{j^{th}$ word in i^{th}sentence$\}$
Bag of Phrases	$\{0,1\}^{\lvert V \rvert}$	$\mathbf{1}\{j^{th}$ phrase in i^{th}sentence$\}$
Word Sentiment Counts (WSC)	$\mathbb{Z}^5$	# words with sentiment j in i^{th}sentence
Word Features (WF)	$\mathbb{Z}^{3n_i}$	word index in dictionary V of j^{th} word in i^{th}sentence tf-idf of j^{th} word in i^{th}sentence sentiment of j^{th} word in i^{th}sentence

(a) Summary table of vector-based features.

Model Name	Bag of Words	Bag of Phrases	WSC	WF	Dictionary indices	Negation	Tree
Sentiment Sum			✓			✓	
Naive Bayes	✓	✓	✓		✓		
K-Nearest Neighbors	✓	✓	✓	✓			
Softmax Regression	✓	✓	✓	✓			
SVM	✓	✓	✓	✓			
Tree prediction						✓	✓

(b) Summary table of models.

Table 2: Summary of features and models.

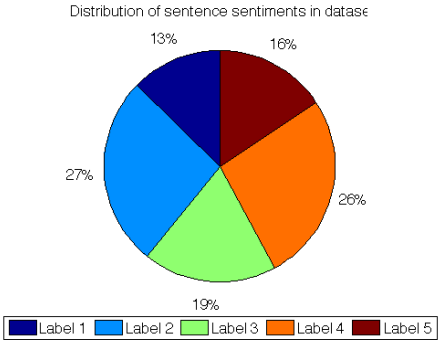

Figure 2: Distribution of sentence sentiment labels across dataset.

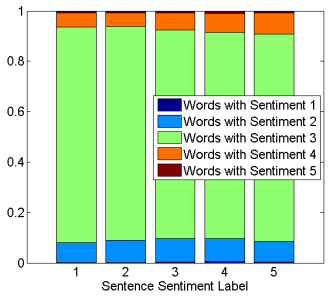

Figure 3: Histogram of sentiment labels for each sentiment label.

some domain, where the length of the sentence is n_i words, and $x_j^{(i)}$ is the j^{th} element in the feature vector $x^{(i)}$. Our dictionary of words and phrases are V_W and V, respectively. A summary of these features is in Table 2a.

Bag of Words, Bag of Phrases In the bag of words representation a vector of length equal to the dictionary size is populated with indicators for the presence of each word within a sentence. This approach takes into account neither individual word sentiment nor positioning of words, but it is very simple to implement. The expansion to a bag of phrases model adds sentence fragments to the model. The sentence fragments incorporate context and phrasing that were ignored by the bag of words model. The bag of phrases model is limited by the level of overlap between phrases used in the training and test sets.

Word Sentiment Counts (WSC) This model represents a sentence as a 5 element vector where each element indicates the number of sentence words with the corresponding sentiment label. This model assumes that individual word sentiments are correlated with the sentiment of the overall sentence. The distribution of word sentiments remaining nearly constant with varying sentence sentiment as observed in Figure 3 indicates that this is likely to be a poor assumption.

Words with Features (WF) This feature vector aggregates three different vector sentence representations. The first is the dictionary index of each word in the sentence. With the dictionary of words this allows complete reconstruction of the sentence. This means that sentence structure and context are incorporated, although a linear predictor is unlikely to be able to predict more than a small portion of the structure in this representation. The second component is the term frequency-inverse document frequency for each word in the sentence. This metric provides an indication of the level of significance of the word within the corpus. The last component is the sentiment label of each constituent word of the sentence.

3.2 Models

The most basic estimation of sentence sentiment comes from a vector sum of the sentiments of the words in the sentence. Shifting the result such that a vector of neutral words remains neutral the majority vote of non-neutral terms determines the estimated sentence sentiment. The conclusion of

Figure 3 is that this should be a poor estimate of the sentence sentiment as sentences of all sentiments contain non-neutral words with nearly constant distributions. Positive words are generally not used to buoy the sentiment of sentences with overall negative sentiment. Instead they are generally altered with control words or used as a contrast to emphasize the negative sentiment of the sentence. Movie reviews in particular make use of flamboyant and sarcastic phrasing to express negative sentiments. The most straightforward set of control words are inversion of which "not" and the contraction suffix "n't" are typical examples. Incorporating negation in a model building sentence sentiments from word sentiments should improve performance. Negation in a sentence generally alters a sub-section of a sentence rather than applying to all words within the sentence. Localized negation is not compatible with linear models and must either be used with simple models such as vector sum of sentiments or be introduced as a derived modification to the feature vector before modeling.

Most negation has the position of the negating word as a well defined initial boundary for the extent of the negation. Determining the final extent of the negation is much a difficult problem [2]. We have used the approximation that negation runs until the end of the sentence but also allow multiple negations to accumulate and continue to flip the sentiment interpretation.

We used Naive Bayes, K-Nearest Neighbors, and Softmax regression to test both our dictionary feature set and our word sentiment feature set. For K-Nearest Neighbors, we set k to be 20% of the training samples; for each test vector, the model selects the k-nearest training vectors, selected by hamming distance, and outputs the mode (most frequent) of the training sentiments. We ran softmax regression to converge within a change of 0.05, which was 3820 iterations (alpha = 0.1) on the full phrase dictionary bag of phrases and 705 iterations (alpha = 0.01) on the word sentiment vector. We implemented these models in Matlab.

In the below equations, m is the size of our training set, and $g(x^{(i)}) = s$ describes a hypothesis function g that gives the ith sentence a sentiment label s, as a function of the feature vector $x^{(i)}$.

Sentiment Sum We used a vector sum of word sentiments as our baseline model, using the WSC model:

$$g(x^{(i)}) = \sum_{j=1}^{5} (j{x_j}^{(i)} - 3) + 3$$

Negation tracking detects a negating word and inverts the sentiment of all subsequent words.

Naïve Bayes (NB) We used Bag of Words, Bag of Phrases, and a modified version of our feature vectors which ignored the frequency products and word sentiments and instead focused only on dictionary indices of the sentence. Our Naïve Bayes model used a Bernoulli event model for each of the 5 sentiment labels and the most probable label was selected.

$$p(g(x^{(i)}) = s|x^{(i)}) = \frac{(\prod_{j=1}^{m} p({x_j}^{(i)}|g(x^{(i)}) = s))p(g(x^{(i)}) = s)}{\sum_{\hat{s}}(\prod_{j=1}^{m} p({x_j}^{(i)}|g(x^{(i)}) = \hat{s}))g(x^{(i)}) = \hat{s})}$$

K-Nearest Neighbors (KNN) We used Euclidean distance to find the k closest training samples and returned the most common sentence sentiment label, and tested/trained on all features.

$$D(x, x^{(i)}) = \|x - x^{(i)}\|_2^2$$

$$h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)}; \theta) \\ p(y^{(i)} = 2|x^{(i)}; \theta) \\ p(y^{(i)} = 3|x^{(i)}; \theta) \\ p(y^{(i)} = 4|x^{(i)}; \theta) \\ p(y^{(i)} = 5|x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{5} e^{{\theta_j}^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ e^{\theta_3^T x^{(i)}} \\ e^{\theta_4^T x^{(i)}} \\ e^{\theta_5^T x^{(i)}} \end{bmatrix}$$

$$g(x^{(i)}) = \text{mode}_{x^{(p)}}(y_p) : x^{(p)} \in \{k \text{ nearest samples}\}$$

Softmax regression with l1-regularization We tested and trained softmax regression on all features.

$$\max_\theta \sum_{i=1}^{m} \sum_{j=1}^{5} \mathbf{1}\{y^{(i)} = j\} \log p(y^{(i)} = j|x^{(i)}; \theta) - \frac{\lambda}{2} \sum_{p=1}^{5} \|\theta_p\|_2^2$$

The gradient ascent update equation is as follows:

$$\sum_{i=1}^{m} [x^{(i)}(\mathbf{1}\{y^{(i)} = j\} - p(y^{(i)} = j|x^{(i)}; \theta)] - \lambda\theta_j$$

SVM We used liblinear 1.95 to train SVM models for our feature vectors.

Tree prediction Elements of node recombination matrix are defined by the average sentiments of nodes in the training set with child labels corresponding to the indices of the matrix element. Negation was included in this model by adding a negation label to the set of sentiment labels. Negation is primarily right combining and not all trees separate phrases such that negations occur as the left child of a node so the negation persists in instances where the negation occurs on a right child.

$$M_{i,j} = \frac{\sum_{k=1}^{m} \sum_{N \in Nodes(x^{(k)})} s(N)\mathbf{1}\{s(N_{left}) = i\}\mathbf{1}\{s(N_{right}) = j\}}{\sum_{k=1}^{m} \sum_{N} \mathbf{1}\{s(N_{left}) = i\}\mathbf{1}\{s(N_{right}) = j\}}$$

4. RESULTS

The simple baseline model of vector sum of sentiments provided better results than many of our more expressive models. This is particularly true for polarity error as shown in Table 3. Test error and train error are determined by correct labeling from the 5 sentiment labels. Polarity error is determined by the polarity of the predicted sentiment being correct for non-neutral sentences. The expected gain in performance of the vector sum of sentiments method from negation tracking did not show up. There are 357 sentences that include negations in our test set of 2210 sentences so the improvement from negation tracking should be small but not

Model	Features	Test Error	Train Error	Polarity Error
Sentiment Sum	WSC	0.67	0.67	0.25
	Negation	0.67	0.67	0.26
Naïve Bayes	Bag of Words	0.77	0.26	0.42
	Bag of Phrases	0.75	0.13	0.26
	WSC	0.72	0.75	0.46
	Dictionary indices	0.62	0.46	0.37
KNN	Bag of Words	0.77	0.73	0.82
	Bag of Phrases	0.77	0.73	0.82
	WSC	0.77	0.73	0.82
	WF	0.71	0.74	0.82
Softmax	Bag of Words	0.73	0.22	0.52
	Bag of Phrases	0.59	0.10	0.39
	WSC	0.65	0.64	0.41
	WF	0.63	0.64	0.39
SVM	Bag of Words	0.72	0.20	0.41
	Bag of Phrases	0.60	0.00	0.23
	WSC	0.62	0.63	0.26
	WF	0.62	0.62	0.27
Tree prediction	Tree	0.65	0.64	0.41

Table 3: Table of results.

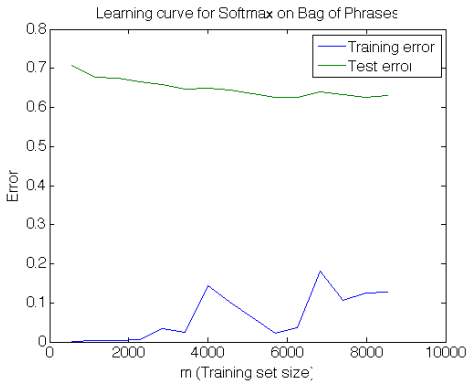

Figure 4: Test and training error vs. training set size for softmax regression on Bag of Phrases.

insignificant. There are several possible explanations for the lack of improvement. The first is that our choice to negate until the end of the sentence is a poor estimate of the proper negation extent. The other good estimate of the extent of negation is to negate until the next punctuation mark, but the benefit of this form of negation tracking is also reported to be insignificant [2]. Another explanation for the poor performance is that negation in sentences from movie reviews is used in hyperbolic phrases and combined with other rhetorical devices such that the effect of negation tracking alone is insignificant. The conclusion from either of these explanations is that simple negation tracking does not improve sentiment labeling.

The Naïve Bayes method provides relatively poor performance for most of our feature vectors that is only slightly better than a random guess for 5-bin sentiment accuracy. Evaluating the model on the training set provides dramatically improved results, potentially indicating an overfitting problem. Reducing the training set of phrases to only a set of words reduces the discrepancy between the training and test error rates, but also results in overall worse performance. The doubling of training set error for bag of words compared to bag of phrases makes it clear that it is longer phrases that are key to the prediction strength of this naive Bayes model, but training the longer features requires a training set that shares these longer features with the test set. With a much larger dataset this method may be able to overcome its lack of structural awareness, but for this dataset it does not seem practical beyond use as a baseline for comparing other methods. Naïve Bayes with a feature vector of dictionary indices provides noticably better 5 level accuracy than vector sum of sentiments model but provides a significantly higher polarity error.

Softmax regression provides good performance with all feature types other than bag of words and the bag of phrases features provide the best 5 level test error of all our models. The training set error for bag of words and bag of phrases is dramatically lower than the test error rate. The features in these two scenarios are derived from the words and phrases present in the training set and many of these are unique in the dataset. This causes significant overfitting to the training set but only the portion of these overfit features that also appear in the test set are realized as overfitting errors.

SVM provides similar 5 level error rates for the different feature types as softmax regression. The bag of phrases feature set once again provides the best performance although it is slightly inferior to softmax. The advantage to SVM comes from the significantly lower polarity error. The overfitting of the bag of phrases feature set is most apparent in SVM where a 0% training set error rate is attained.

The relatively high polarity error for the best Naïve Bayes and softmax regression models indicate that these are relatively poor models for general application. When they do make a mistake on the 5 level sentiment for a non-neutral word they make a large number of big mistakes as reflected by their roughly 40% polarity error rates.

KNN provides very poor results for all of the features tested. The polarity error rate is higher than the 5 label error rate as the polarity error rate is evaluated on non-neutral sentences only. The high error rate of KNN is likely due to the Euclidean distance metric chosen; for high dimension binary vectors such as produced by the bag of words and bag of phrases models, the distance between sentence vectors is large and close to randomly distributed.

The bag of phrases feature set provides the best performance for the SVM and softmax regression models but it is not consistent as the error rate with both KNN and Naïve Bayes is near 80%. The reduced feature set of bag of words provides uniformly inferior performance to bag of phrases. For SVM and softmax regression this difference due to the context and phrasing that is incorporated by the bag of phrases model provides a dramatic improvement in test error rate of at least 15%. WF is much less variable with the training

model used as it never has great performance but it also never has the worst performance.

We attempted to use Principal Component Analysis (PCA) in order to select the most relevant features from our models that exhibited overfitting. We did not get any benefit from this which is likely due to the unusual features being unlikely to appear in the test set.

The learning curve for softmax regression show in Figure 4 shows a large separation between test and training error. The bag of phrases feature set depends heavily on those phrases that are represented in the training set and the overlap of those phrases with the test set. With a dramatically larger training set the training and test errors would converge to some intermediate value. No matter how large the training set is, however, such a model will be unable to properly decompose a test sentence with new and original phrasing. A model that groups similar words and then trains their phrasing patterns has a much lower error rate limit and requires much less training data.

The best literature result provides a 54.3% error rate for sentences for 5 sentiment labels [1]. Our best result of 59% is close but also notably below this level.

Our initial intention was to compare sentiment analysis models that used a vector sentence representation to recursive tree recombination models. We were able to conclude that vector sentence representations are noticeably less expressive in terms of sentence structure and word context. We thus started looking at applying models to tree reconstruction. The general reconstruction problem is expressed by training an $W \times W$ matrix where W is the number of words in the training set dictionary. W is 22348 in our dataset which leads to a matrix that is much too large to be well estimated with our set of training sentences. The neural network based approach introduced by Socher et al. appears to be a very promising way to reduce the recombination problem to a manageable form [1]. We took an alternate approach by reducing the feature set of the recombination matrix. The reduction of the matrix to a set of 5 sentiment labels plus a negation label makes the model very similar to the vector sum of sentiments model and the resulting performance is also similar.

5. CONCLUSION

We found that a bag of phrases feature set with an SVM model provided the best results among vector based sentence models. While negation tracking did not provide any benefits in our experiments, negation is fundamental to the interpreted sentiment of many of our test sentences. A better method of handling negation that potentially also accounts for contrastive sentence structures will certainly improve performance. The additional flexibility of tree based representations appears to be the best future path for sentiment analysis.

6. FUTURE

If we were to expound on this project, we would try implementing the following list.

- Develop additional features for use in tree reconstruction
- Explore deep learning for tree reconstruction of sentences as pioneered by Socher et al.
- Apply classification algorithms to phrases
- Add negation tracking to linear models
- Find a better distance metric for KNN
- Build up from a model of word compositions and build towards sentences rather than decomposing sentences

References

[1] Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality. *Empirical Methods in Natural Language Processing* (2013).

[2] Wiegand, M., Balahur, A., Roth, B., Klakow, D., and Montoyo, A. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing* (July 2010).