International Conference on Semantic Computing

If We Want Your Opinion

Daniel M. Bikel and Jeffrey Sorensen
IBM T. J. Watson Research Center
Yorktown Heights, NY
{dbikel,sorenj}@us.ibm.com

Abstract

Sentiment has traditionally been considered a "deep" attribute of writing, often requiring the interpretation of figurative language to uncover the writer's intention. The natural language processing community has become increasingly interested in detecting, through automatic means, the expression of opinions and measuring the intensity of emotions held by the writer. Despite the depth and abstraction often associated with expressions of sentiment, we apply strictly lexical analysis to the opinions expressed about books and find that machine learning techniques are capable of resolving even fine-grained distinctions between opinions. Using an averaged perceptron classifier trained using a word subsequence kernel, we achieve an accuracy of 89% when distinguishing between 1- and 5-star reviews. Further, this same model yields significant separation when scoring intermediate reviews—making distinctions even human annotators find difficult. We detail the collection of data for supervised training and present the results of our sentiment classifier along with some discussion about why we believe this approach to be effective.

1 Introduction

What movies should I see? What music should I be listening to? What books should I read next? Assuming a bounded amount of time to invest in being entertained, there is widespread interest in pruning the ever growing set of potential material for viewing and listening. Professional movie critics are only one example of the army of people paid by media companies to view and mete out opinions that hopefully guide us in our choices.

With the explosion of blogs, and with commercial sites like Epinions.com (owned by Ebay.com) and Amazon.com, Inc. featuring collected reviews from their own customers, now more than ever we are inundated with opinions, not all of them welcome. Being able to synthesize a single numerical score for each review is an essential step in developing an aggregated review. While such a score is itself of debatable value, the popularity of sites like Metacritic[1] and Rotten Tomatoes[2] demonstrate the demand for this degree of radical oversimplification. Furthermore, the potential of an opinion-based economy to steer customers to products could truly revolutionize our current advertising-centric approach to selling. In fact, at least one study [12] has demonstrated a causal link between customer-provided reviews and customer purchasing behavior.

Being able to read and analyze, systematically, criticism and opinion is an ambitious undertaking. In this work we begin with a much more modest attempt to approximately characterize a review as positive or negative using only simple lexical features. In the following sections we review the earlier publications dealing with customer review data, our methods for collecting training and testing material, the design of our algorithm, and the results of our experiments both in estimating review valence and usefulness.

[1] `http://www.metacritic.com/`, a division of CNET Networks, Inc.

[2] `http://www.rottentomatoes.com/`, a division of News Corp.

0-7695-2997-6/07 $25.00 © 2007 IEEE
DOI 10.1109/ICSC.2007.81

2 Previous Work

Much of the effort in sentiment analysis in recent years has focused on the problem of binary classification of text into positive or negative opinions. Such binary classification of opinion blurbs was the goal of Turney [16], who tested a model on 412 reviews from the Epinions.com website. Turney performed part-of-speech tagging on each review, then used a pointwise mutual information/information retrieval (PMI-IR) model to each phrase to determine its "semantic orientation" (positive or negative), then took the average semantic orientation of all the phrases in the review. This approach yielded 74% accuracy overall types of reviews, ranging from 66% accuracy for movie reviews to 84% accuracy for automobile reviews.

More recently, Pang and Lee [14] have explored the more fine-grained assignment of the *number* of stars to each review. In that work, the authors recast the n-ary classification task to one of *metric labeling*, attempting to exploit the fact that reviews whose star ratings are similar are similar themselves. After performing pilot studies with humans to confirm star differences can be accurately annotated, the authors present three different approaches: one-versus-all n-ary classification, linear regression and a Markov random field that explicitly encodes the relationship between the distance function of the star ratings and the distance function between reviews. Pang and Lee attempt to classify individual sentences within reviews and then calculate the percentage of positive sentences (the number of positive sentences divided by the total number of subjective sentences).

Goldberg and Zhu [8] build on Pang and Lee's work by incorporating unlabeled data, using a graph-based semi-supervised learning algorithm. Their transductive learning setting showed performance improvements over the work of Pang and Lee, but only when the amount of labeled data was small. Mullen and Collier [13] use a support vector machine to combine feature sets from several previous efforts to good success on both Pang and Lee's Epinions.com data set and a new data set of music reviews.

It is also worth noting that the measure of sentiment is not only of theoretical interest. In [7], using the methodology of Pang and Lee, a classifier built to measure the ratio of *subjective* to *objective* content in product reviews was correlated with both product sales and review helpfulness.

In other efforts, Hu and Liu [9] use a 12-stage pipelined architecture, including part-of-speech tagging, feature identification and sentence opinion extraction, in order to produce customer review summaries. In another, related effort in sentiment analysis, Kim et al. [10] attempt to predict review *helpfulness*. In that work, the researchers threw a variety of features into their model, including IR-based lexical features, syntactic (read: part-of-speech) features and "semantic" features (lexical features that are closely related to the opinion of a review, or to a description of the item being reviews).

In nearly all previous work, there is a word- or sentence-level model, the results of which are averaged across all words/sentences/n-grams in order to produce a single model output for each review. By contrast, we use subsequence kernels [11] to have our model explore the space of all "gappy n-grams" in the reviews, in our case, up to four words in length. Our intuition is that the feature space implicitly captured by subsequence kernels is sufficiently rich to obviate the need for explicit knowledge engineering or modeling of word- or sentence-level sentiment. Finally, by using a voted perceptron model [6], we still gain the benefits of locally-weighted learning, where we may explicitly model the distance between differently-rated reviews.

3 Training Data

Using the Web Services API [1] provided by Amazon.com, we obtained book reviews for books in a number of arbitrarily chosen categories. The search was done by providing the single keyword, then for each book in that category we would request all of the reviews associated with each book. We would terminate the search of a keyword category once ten pages of books with no reviews are returned by Amazon.com. From each review, the summary text, the body text, the star rating, and the number of yes and no "useful" votes were all extracted for training purposes.

Table 1 shows the complete list of categories and the number of reviews for each category and star rating. Amazon.com reviews are highly biased towards

positive reviews, with 5-star reviews often outnumbering 1-star reviews by 10-to-1 or more. Given that reviews are returned essentially in a random order, we took the first 90% of the reviews in each category and assigned it to our training set, and the remaining 10% constituted our blind, held-out test set.

	Number of stars					
Keyword	1	2	3	4	5	Total
art	582	431	661	1693	5600	8967
drugs	260	172	316	788	2716	4252
genetics	139	114	233	501	1385	2372
history	703	506	912	2407	6520	11048
homeopathy	32	15	20	66	323	456
israel	326	149	224	524	1783	3006
java	793	569	590	1132	2118	5202
love	847	668	983	2288	7590	12376
magic	581	516	1055	2529	7279	11960
murder	778	624	1120	2345	4493	9360
music	414	401	682	1735	5733	8965
race	395	282	392	1057	2970	5096
religion	721	435	745	2024	8758	12683
scientology	37	4	5	24	139	209
sex	888	513	722	1692	5606	9421
Total	7496	5399	8660	20805	63013	105373

Table 1. Total data harvested.

It is important to note that there is no oracle for these reviews; that is, each reviewer assigns a star rating according to their own judgment. Whereas Pang and Lee [14] primarily investigated training separate classifiers on the reviews of well-known movie critics to avoid the "author calibration problem", we are training a single classifier on the subjective star ratings of a large and disparate array of individuals.

4 Pre-processing

Each book review on Amazon.com consists of a star rating, a short text summary, and a longer textual review. The text reviews can vary widely in length from a simple "great book!" to lengthy descriptions that can even include extensive quotes from the material being reviewed. In addition, readers of the reviews are allowed to vote "yes" or "no" when asked if the review was helpful. We discuss the use of these votes in §8.

For our algorithm, we must represent each review s as a string chosen from a alphabet Σ. This alphabet, in our case, is the set of all lexical items that occur in the training set with count four or higher. We use a straightforward regular expression to tokenize, crucially allowing single quotes to occur inside tokens, in order to capture contractions like "don't" which are frequent in reviewers' texts. For our training set, this results in a 62K vocabulary. Primarily for computational reasons we discard all words not in the dictionary and have truncated reviews longer that 256 words.

5 Word Sequence Kernels

Many machine learning algorithms require a function for comparing examples for classification with the labeled instances in the training set. In our case, we would like a function that permits an efficient exploration of the exponentially-large feature space of word subsequences. We have therefore chosen a *word sequence kernel*. Word sequence kernels [2] are a variant *string kernels* as presented in [11], but with the review text represented as a sequence chosen from an alphabet of whole words. In [2], word sequence kernels were applied to document classification.

Word sequence kernels of order n are a weighted sum over all possible word sequences of length n that occur in both of the strings being compared. For a kernel of order 2, the subsequence of words {"great", "book"} occur in both the texts "great reference book" and "if you need a great cure for insomnia try reading this book" but within different length spans. Because they count all word sub-sequences that are common between two strings, these kernels are also known as rational kernels [4], or, more colloquially, gappy n-gram kernels.

Mathematically, the word sequence kernel is defined as

$$K_n(s,t) = \sum_{u \in \Sigma^\star} \sum_{\mathbf{i}:s[\mathbf{i}]=u} \sum_{\mathbf{j}:t[\mathbf{j}]=u} \lambda^{(i[n]-i[1]+1)+(j[n]-j[1]+1)} \quad (1)$$

where λ is a kernel parameter that can be thought of as a gap penalty. $\mathbf{i}$ refers to a vector of length n that consists of the indices of string s that correspond to the subsequence u. And, the value $i[n] - i[1] + 1$ can be regarded as the total length of the span of s that constitutes a particular occurrence of the subsequence

u. Following Rousu et al. [15], we combine the kernels of orders one through four through an exponential weighting

$$K(s,t) = \sum_{i=1}^{N} \mu^{1-i} K_i(s,t). \quad (2)$$

For all of the experiments presented below, we fixed the parameters $N = 4$, $\lambda = 0.8$, and $\mu = 0.5$.

6 Voted Perceptron

Among the many choices for classifiers, we chose to use the voted perceptron algorithm [6], for several reasons. This algorithm is a large-margin linear classifier, falling in the general family of locally-weighted learning algorithms. As such, it gives us the flexibility to incorporate the star ratings fairly directly, by projecting them onto the [-1,1] interval as numerical labels on the training instances. Moreover, it is readily amenable to the use of kernel functions, allowing the exploration of a much larger feature space without much additional computational cost. Finally, it has been experimentally shown to have fast convergence.

$j \leftarrow 1; \mathbf{v}_0 \leftarrow \mathbf{0}; c_0 \leftarrow 0$
repeat {up to T times}
  for i from 1 to n **do**
    if $\mathrm{sgn}(\mathbf{v}_j \cdot x_i) = \mathrm{sgn}(y_i)$ **then**
      $c_j \leftarrow c_j + 1$
    else
      $\mathbf{v}_{j+1} \leftarrow \mathbf{v}_j + y_i \mathbf{x}_i$
      $c_{j+1} \leftarrow 1$
      $j \leftarrow j + 1$
    end if
  end for
until convergence or Tth iteration a list of weighted perceptrons
$\langle (\mathbf{v}_1, c_1), \ldots, (\mathbf{v}_j, c_j) \rangle$

Figure 1. The voted perceptron algorithm (primal form).

Figure 1 illustrates the algorithm in its primal form, where the input is a series of n vectors $\mathbf{x}_i \in \mathcal{X}$ with associated labels $y_i \in \mathcal{Y}$ and a number of epochs T. Crucially, labels for positive examples should be positive (typically 1.0) and those for negative examples

$j \leftarrow 1; \boldsymbol{\alpha}_0 \leftarrow \mathbf{0}; c_0 \leftarrow 0$
repeat {up to T times}
  for i from 1 to n **do**
    if $\mathrm{sgn}(\sum_{k=1}^{n} \alpha_{j,k} y_k (\mathbf{x}_k \cdot \mathbf{x}_i)) = \mathrm{sgn}(y_i)$ **then**
      $c_j \leftarrow c_j + 1$
    else
      $\boldsymbol{\alpha}_{j+1} \leftarrow \boldsymbol{\alpha}_j; \quad \alpha_{j+1,i} \leftarrow \alpha_{j+1,i} + 1$
      $c_{j+1} \leftarrow 1$
      $j \leftarrow j + 1$
    end if
  end for
until convergence or Tth iteration a list of dual-form weighted perceptrons
$\langle (\boldsymbol{\alpha}_1, c_1), \ldots, (\boldsymbol{\alpha}_j, c_j) \rangle$

Figure 2. The voted perceptron algorithm (dual form), where $\alpha_{i,j}$ is the jth component of the ith $\boldsymbol{\alpha}$ vector.

are negative (typically -1.0). As [6] note, and as described in detail in [5], there is a dual form of the algorithm in which every perceptron is a weighted sum of a subset of the training instances, illustrated in Figure 2. This representation is typically much more compact than the primal form, and also illustrates how a the voted perceptron essentially implements a *bagging model*, where each different weighted sum of instances is some view of the training data. Finally, the fact that the dual form only computes dot products between training instances lets us substitute an arbitrary kernel function for the dot product function, implicitly exploring a much larger feature space. Also, for all of the models discussed in this paper, we train on only the extreme reviews (1 and 5 star ratings), even though we test on all reviews.

7 Predicting Star Rating

For prediction, we use the weighted average of the scores returned by the perceptrons (also known as an *averaged perceptron*). These are unnormalized scores, meaning the absolute values are only bounded by our imposed hard word limit of 256.

Our model, despite being trained only on the extreme one and five star reviews, forms an excellent continuum over reviews with intermediate star ratings, as shown in Figure 3. It is rare that we see such behavior associated with lexical features which are typ-

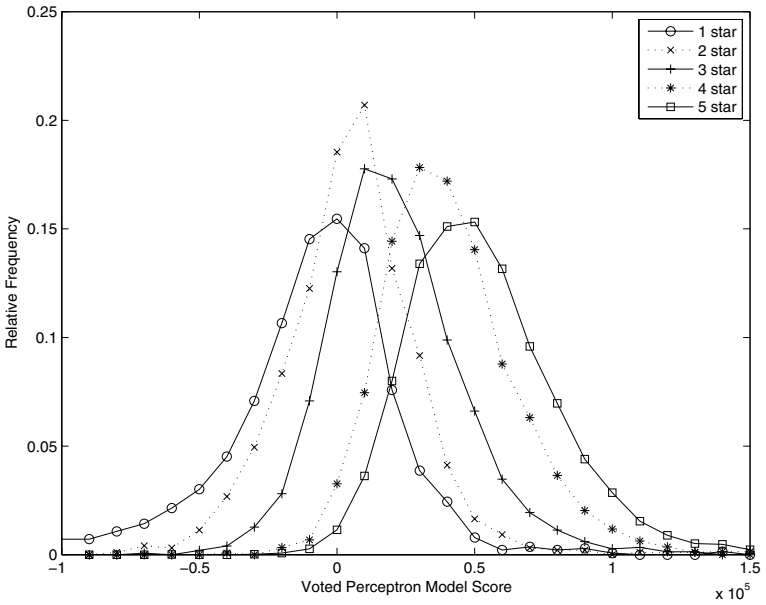


Figure 3. Distribution of voted perceptron model scores by number of stars

ically regarded as discrete and combinatorial. Finally, we note that the voted perceptron is making distinctions that humans found difficult in [14].

While the trend among these predicted distributions is visually obvious, we checked the significance of the observed differences via a series of paired t-tests. Figure 4 illustrates the results. Modulo the outlying points in the 3- and 4-star distributions, we achieve significant distinctions among the means of the various pairs. As Figure 4 indicates, we re-ran three of the t-tests involving the 3- and 4-star distributions removing outliers. We justify these additional tests by noting that our entire data set does not contain "gold-standard" truth; rather, it contains the subjective star ratings of a disparate set of on-line shoppers.

The receiver operating characteristic (ROC) plots of Figures 5(a) and 5(b) show the the ability of our voted perceptron model to partition the reviews. In addition to testing our subsequence kernel model, we tested a simpler model, where each review was treated as a vector representing a bag of words, and the inner product of two such vectors is equivalent to the term-frequency cosine distance. This kernel is a fourth-order polynomial expansion kernel

$$K_{\text{poly}}(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1 \cdot \mathbf{v}_2 + 1)^4 \qquad (3)$$

to be roughly comparable with our fourth-order subsequence kernel.

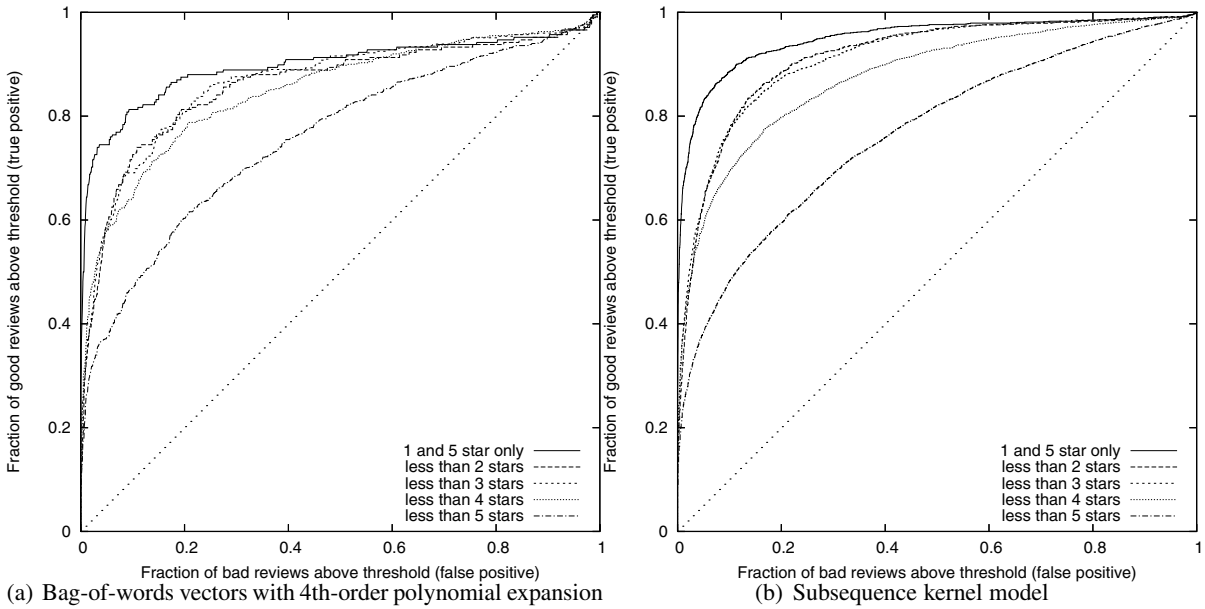
	2	3	4	5
1	1	1	$0^{\dagger}$	1
2		0	$0^{\dagger}$	1
3			0	$0^{\ddagger}$
4				1

†Removing 4 outlier points ($\geq 3\sigma$ from the mean) from the 4-star reviews yields a statistically significant t-test. ‡Removing 2 outlier points from the 3-star reviews yields a statistically significant t-test.

Figure 4. Paired t-tests for pairs of distributions shown in Figure 3. All positive results are at p-value of ≤ 0.05.

When tested only on one and five star reviews we achieve an accuracy of 89% for the equal error rate condition. Of couse, in most real-world test scenarios a model would have to deal with reviews of all possible star ratings, not just the extremes. When we split the reviews into two classes, 1-star and more than 1-star, we were able to identify the "bad" reviews with 85% accuracy.

We perform a baseline comparison using a conventional support vector machine classifier trained using LIBSVM [3], using the same review data with a bag



(a) Bag-of-words vectors with 4th-order polynomial expansion kernel

(b) Subsequence kernel model

Figure 5. ROC curves for polynomial kernel (a) and subsequence kernel (b). "1 and 5 star only" indicates testing only on 1- and 5-star reviews. "Less than two stars" indicates testing on all reviews, with classification of 1 star versus not 1star, etc.

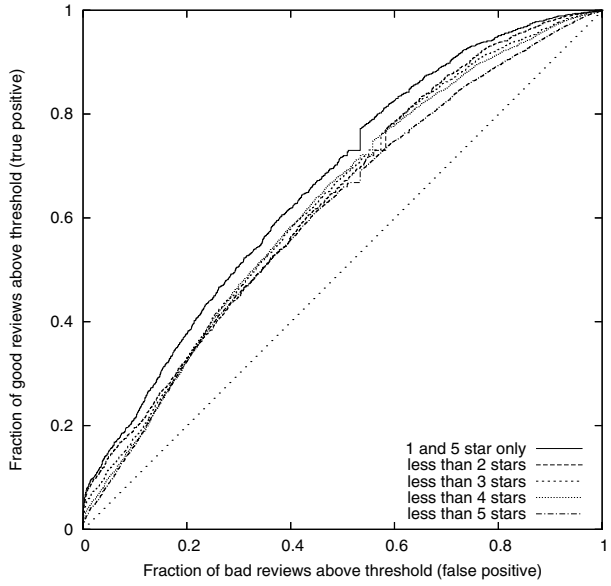


Figure 6. ROC curves for baseline SVM bag-of-words model.

of words representation and a simple linear kernel. Figure 6 shows the ROC plots of this baseline model. While better than chance, the classifier is unable to distinguish reviews with accuracies better than 61% under any conditions.

Another question that we seek to address is whether or not the voted perceptron sentiment classifier is capable of correctly classifying across domains. To test this, we built a classifier using only the data from the categories race, Java, Israel, drugs, and genetics (our *five-category* data set). We then tested this classifier on the remaining four categories using only the 1 and 5 star reviews. The resulting ROC curve is shown in Figure 7. While there is some variation in the performance, the word sequence kernel voted perceptron appears to be quite effective at generalizing across unrelated domains.

8 Predicting Utility

In addition to allowing customers to review books, the reviews themselves are subject to another meta-level of criticism. Customers may vote upon reviews

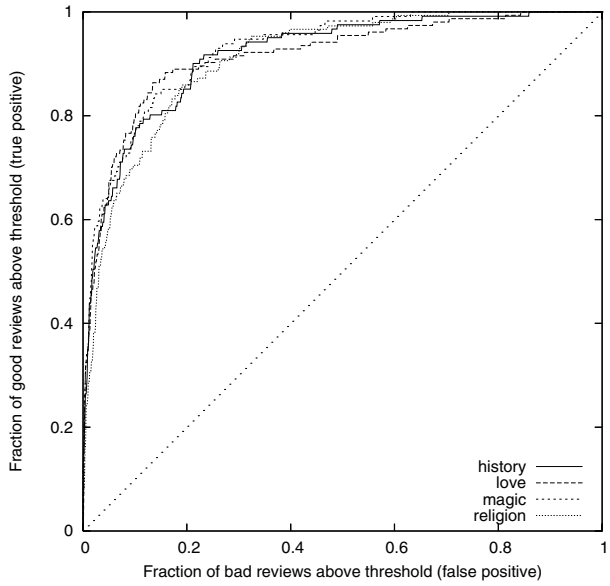


Figure 7. ROC curves on held-out, out-of-genre results.

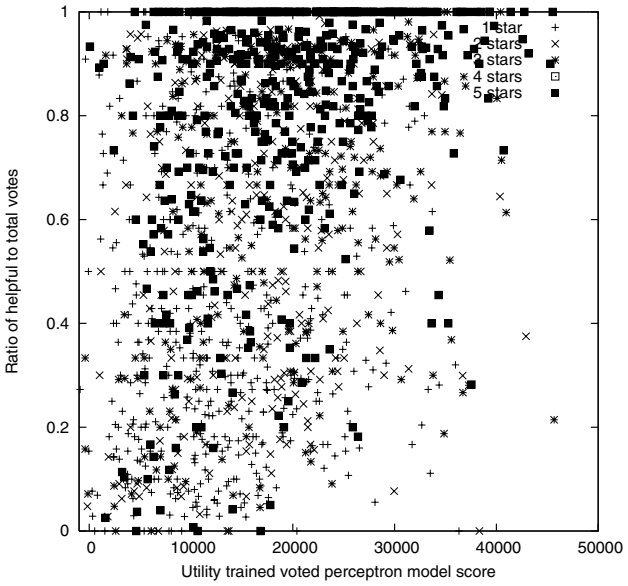


Figure 8. Scatterplot of utility quotient versus model score.

with a binary yes or no in response to the question "Was this review helpful to you?" One might presume that this means that the reader of the review bought the product, evaluated it themselves, and generally agreed with the content of the review and later returned to the site to select "Yes." However, it is more likely that customers evaluate the surface of the review and evaluate how helpful the review is at making their decision to buy the product or not.

In the case of book reviews, there are a great many reviews that say things like "couldn't finish the book" or "didn't read," and these phrases are typical of those found in reviews generally labeled as not helpful. In many respects, the ability to quickly judge whether or not a review is likely to be helpful *a priori* might be more valuable, if it can be done automatically, than being able to sense the sentiment.

Presumably, the surface features that distinguish both helpful and non-helpful reviews are different than those that give a book a positive or negative. To test our ability to provide this classification, we trained a voted perceptron on the five-category data using only those reviews that were voted unanimously helpful or not-helpful. Of the 32K reviews in the five-category training data, 2.7K were unanimously not-helpful, and 13.6K were unanimously helpful. Figure 8 shows the somewhat disappointing results. While the perceived emptiness of the lower-right quadrant suggests a subtle yet perceptible positive trend, there is clearly a lack of separation. We suspect that a review's utility is a more subtle metric than its star rating. In the plot we distinguish the points by number of stars; however, the classifier does not have access to this feature. As [7, 14] show, incorporating information from the product description, and measuring the ratio of subjective to objective comments in reviews is a predictor of review helpfulness. Our lackluster utility results are perhaps not altogether surprising, given that our classifier does not have access to the product description. We intend to incorporate such information in future utility experiments.

9 Conclusion

Sentiment analysis in general, and review rating prediction in particular, has application well beyond the initial sphere of retail sales and marketing. Indeed, it forms a subset of the more general problem of text classification, and as such, it may be brought to bear on newswire text for financial analysis, political commentary for sentiment of the populace, and undoubtedly many more applications. In this paper,

we have explored the novel combination of two well-known techniques: the subsequence kernel and the averaged voted perceptron. The subsequence kernel provides the richness of "gappy n-gram" features that appear partially to obviate the need for knowledge-rich methods for modeling sentiment in individual sentences. The voted perceptron provides a convenient locally-weighted learning model that can learn a function from word sequences to the real numbers that correlates with the metric of the input labels—even when only trained on the extreme-labeled examples (the 1- and 5-star reviews).

Even though our model has achieved classification accuracy up to 89% on this task, much investigation remains. For example, just because we have initially eschewed explicitly modeling sentiment at the sentence level does not mean additional knowledge sources or features would not help accuracy. Also, following in the footsteps of Pang and Lee, we suspect that eliminating "objective" sentences would also only help accuracy of our models. Finally, we intend to explore training on all instances, not just the extremes, to take advantage of the voted perceptron's ability to model the metric implicit in the input labels.

References

[1] Amazon.com. Amazon web services. `http://www.amazon.com/gp/browse.html?node=3435361`, 2006.

[2] N. Cancedda, E. Gaussier, C. Goutte, and J. M. Renders. Word sequence kernels. *J. Mach. Learn. Res.*, 3:1059–1082, 2003.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *J. Mach. Learn. Res.*, 5:1035–1062, 2004.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[6] Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[7] A. Ghose and P. G. Ipeirotis. Designing ranking systems for consumer reviews: The impact of review subjectivity on product sales and review quality. In *Proceedings of the International Converence on Decision Support Systems*, Kolkata, India, January 2007. Association for Information Systems.

[8] A. Goldberg and X. Zhu. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, pages 45–52, New York City, June 2006. Association for Computational Linguistics.

[9] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177, New York, NY, USA, 2004. ACM Press.

[10] S.-M. Kim, P. Panel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 423–430, 2006.

[11] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. J. C. H. Watkins. Text classification using string kernels. In *Advances in Neural Information Processing Systems*, pages 563–569, 2000.

[12] D. Mayzlin and J. A. Chevalier. The effect of word of mouth on sales: Online book reviews. Yale School of Management Working Papers ysm413, Yale School of Management, Aug. 2003. available at http://ideas.repec.org/p/ysm/somwrk/ysm413.html.

[13] T. Mullen and N. Collier. Sentiment analysis using support vector machines with diverse information sources. In D. Lin and D. Wu, editors, *Proceedings of EMNLP 2004*, pages 412–418, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[14] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[15] J. Rousu and J. Shawe-Taylor. Efficient computation of gapped substring kernels on large alphabets. *J. Mach. Learn. Res.*, 6:1323–1344, 2005.

[16] P. Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424, Philadelphia, Pennsylvania, 2002. Association for Computational Linguistics.