A Hierarchical Classifier Applied to Multi-way Sentiment Detection

Adrian Bickerstaffe and Ingrid Zukerman
Faculty of Information Technology
Monash University
bickerstaffe.adrian@gmail.com, Ingrid.Zukerman@monash.edu

Abstract

This paper considers the problem of document-level multi-way sentiment detection, proposing a hierarchical classifier algorithm that accounts for the inter-class similarity of tagged sentiment-bearing texts. This type of classifier also provides a natural mechanism for reducing the feature space of the problem. Our results show that this approach improves on state-of-the-art predictive performance for movie reviews with three-star and four-star ratings, while simultaneously reducing training times and memory requirements.

1 Introduction

A key problem in sentiment detection is to determine the polarity of sentiment in text. Much of the work on this problem has considered binary sentiment polarity (positive or negative) at granularity levels ranging from sentences (Yu and Hatzivassiloglou, 2003; Mao and Lebanon, 2006; McDonald et al., 2007) to documents (Wilson et al., 2005; Allison, 2008).

This paper considers the more general problem of multi-way sentiment classification for discrete, ordinal rating scales, focusing on the document level, i.e., the problem of predicting the "star" rating associated with a review. This is a supervised learning task involving textual reviews that have been tagged with a rating. Ultimately, the goal is to use classifiers which have been trained on tagged datasets to predict the ratings of untagged reviews.

Typical approaches to the rating scale problem include standard k-way classifiers, e.g., (Pang and Lee, 2005). However, these methods do not explicitly account for sample similarities, e.g., the samples with a "four star" rating being more similar to "three star" samples than to "one star" samples. Consequently, these methods generally do not perform well, while methods which incorporate sample similarity information achieve improved performance (Pang and Lee, 2005).

Sample similarity in the multi-way sentiment detection setting has previously been considered by using Support Vector Machines (SVMs) in conjunction with a metric labeling meta-algorithm (Pang and Lee, 2005); by taking a semi-supervised graph-based learning approach (Goldberg and Zhu, 2006); and by using "optimal stacks" of SVMs (Koppel and Schler, 2006). However, each of these methods have shortcomings (Section 2). Additionally, during the learning process, all approaches employ a set of word/punctuation features collected across *all* rating categories. Hence, the number of features may be very large compared to the number of training samples, which can lead to the model overfitting the data.

The main contribution of this paper is the use of hierarchical classifier trees which combine standard binary classifiers to perform multi-way classification (another approach to reduce multi-class classification to binary classifications is described in (Beygelzimer et al., 2009)). The hierarchical classifier accounts for inter-class similarity by means of tree structures which are obtained using inter-class similarity measures in conjunction with a shortest-spanning algorithm. The tree structures reduce training times since they require only $k-1$ nodes for a k-rating problem. Training times are further reduced by the fact that classifier nodes lower in the tree consider fewer rating classes than those higher up, thereby naturally reducing the number of training samples relevant to lower-level nodes. Additionally, the tree structures offer a means to safely cull irrelevant features at non-root nodes of the tree, thus reducing the dimensionality of the training data for these nodes without loss of information. Our experiments show that our new classifier outperforms state-of-the-art methods on average, achieving improvements of up to 7.00% and 7.72% for three-way and four-way classification problems respectively (Section 4).

2 Related Work

Pang and Lee (2005) incorporated information about label similarities using *metric labeling*, where label relations were encoded via a distance metric. The output of standard k-ary classifiers was then modified such that similar items were more likely to be assigned similar labels. Metric labeling required a label-corrected item-similarity function, which was based on the observation that the *Percentage of Positive Sentences* (*PSP*) in reviews increased as their ratings increased. Notice, however, that item similarity was not incorporated into the first stage of classifier training. Metric labeling adjusted the output of the classifiers only after they were trained without considering rating similarities. Our approach accounts for inter-category relationships from the outset of classifier design, rather than addressing this issue with later adjustments.

Goldberg and Zhu (2006) proposed a semi-supervised learning approach to the rating inference problem in scenarios where labeled training data is scarce. Using a graph-based optimisation approach, Goldberg and Zhu demonstrated that the inclusion of unlabeled reviews in the learning process could produce significantly higher prediction accuracy than predictors trained without unlabeled data. This approach outperformed competing methods when it considered relatively small numbers of labeled samples from the four-category movie review dataset (Pang and Lee, 2005). However, the graph-based method did not perform well when a large number of labeled samples was available. Furthermore, Goldberg and Zhu's graph-based learning method was transductive: new samples could not be classified until they were added to the graph — a problem avoided by our approach.

Koppel and Schler (2006) considered neutral examples, which may express a mixed opinion or may not express any opinion at all, in addition to positive/negative samples. Their experiments showed that neutral examples often did not lie close to the positive/negative decision boundary as previously believed. This gave rise to the idea of "optimal stacks" of SVMs, which were pairwise combinations of binary classifiers that distinguish between two categories for the ternary positive/neutral/negative problem (instead of a single binary classifier trained using only positive and negative samples). The search for an optimal stack is exponential in time. Hence, finding suitable stacks is feasible for the ternary problem, but becomes intractable for larger numbers of categories (in the general case).

Snyder and Barzilay (2007) proposed the "Good Grief" algorithm, which considers multiple aspects of a situation (e.g., a restaurant review that covers service, ambiance and food), and yields a prediction that minimises the dissatisfaction (grief) regarding these aspects. This method significantly outperformed baseline methods and individual classifiers. At present, we do not consider separately different aspects of a review — a task we intend to undertake in the future.

3 Multiclass SVM Classifiers

Since SVMs are binary classifiers, they are often employed for binary sentiment detection. However, as seen above, it is not straightforward to use SVMs for multi-way classification, particularly when there is inter-class similarity.

One might initially expect that a hierarchical SVM classifier could be built using pairwise comparisons of adjacent class labels. However, pairwise comparisons alone do not form a complete classifier, raising the question of how to combine pairwise classifications. The standard techniques to build k-way SVM classifiers are OVA and OVO (Hsu and Lin, 2002), and DAGSVM schemes (Platt et al., 2000). An OVA classifier requires k SVMs for a k-category problem, where the i^{th} SVM is trained using all samples from the i^{th} category versus all other samples. A sample is classified by evaluating all k trained SVMs, and the label of the class which maximizes the decision function is chosen. The OVO scheme trains $\frac{k(k-1)}{2}$ classifiers derived from a pairwise comparison of the target categories. A prediction is made by evaluating each SVM and recording "votes" for the favoured category: the class with the most votes is selected as the predicted category. The DAGSVM scheme builds a *Directed Acyclic Graph* (*DAG*) where each non-leaf node has an SVM that discriminates between two classes. A DAGSVM is iteratively constructed in a top-down fashion by forming a list of all the class labels, and creating a decision node that discriminates between the first and last element of the list. This decision node yields two child nodes, each of which omits one of the two classes that were compared. Each of these nodes then discriminates between the first and last element in its list of classes, and so on. This process continues for each decision path until only one element remains in the list. A sample is classified by successively making decisions down the graph until a leaf node is reached. Like OVO, DAGSVM schemes require training $\frac{k(k-1)}{2}$ decision nodes.

All three techniques suffer from long training times — an issue that is exacerbated by large data sets such as our corpus of approximately 5000 movie reviews (Section 4.1). Additional problems associated with these techniques are: (1) there is no bound on the generalisation error of OVA, (2) OVO schemes tend to overfit, and (3) the performance of a DAGSVM relies on the order in which classes are processed. This order is based on the class labels (rather than similarity between samples), and no practical method is known to optimize this order.

Overfitting also arises when the number of features is very large compared to the number of training samples. In this case, the SVM training process may discover a decision plane that separates the training data well, but performs poorly on unseen test samples. While SVM training algorithms use regularisation to address the overfitting problem, research has shown that a careful reduction in feature vector dimensionality can help combat overfitting (Weston et al., 2003).

A fundamental problem with the above three schemes is that the similarity between samples of nearby classes is not considered. Instead, categories are assumed to be independent. This problem may be addressed by considering SVM regression (SVM-R) (Smola and Schölkopf, 1998), where class labels are assumed to come from a discretisation of a continuous function that maps the feature space to a metric space. However, SVM-R, like the SVM schemes described here, trains on the entire feature set for all the classes in the dataset. In the case of sentiment detection, where words and punctuation marks are commonly taken as features, the sheer number of features may overwhelm the number of training samples, and lead to the model overfitting the data. SVM-R also poses the question of how to quantise the regressor's output to produce discrete class predictions.

3.1 The MCST-SVM Classifier

To address the above problems, we build a decision tree of SVMs that reduces the set of possible classes at each decision node, and takes relative class similarity into account during the tree construction process. We construct the decision tree as a Minimum Cost Spanning Tree (MCST), denoted *MCST-SVM*, based on inter-class similarity measured from feature values (Lorena and de Carvalho, 2005). Each of the decision tree leaves corresponds to a target class, and the interior nodes group classes into disjoint sets. For each internal node in the MCST, an SVM is trained to separate all the samples belonging to classes in its left subtree from those in its right subtree. We use linear SVMs, which have been shown to be effective text classifiers (Pang et al., 2002; Pang and Lee, 2005), and set the SVM parameters to match those used in (Pang and Lee, 2005).[1] Figure 1 contrasts

[1]SVMs are implemented using the C/C++ library `liblinear`, a variant of `libsvm` (Chang and Lin, 2001).

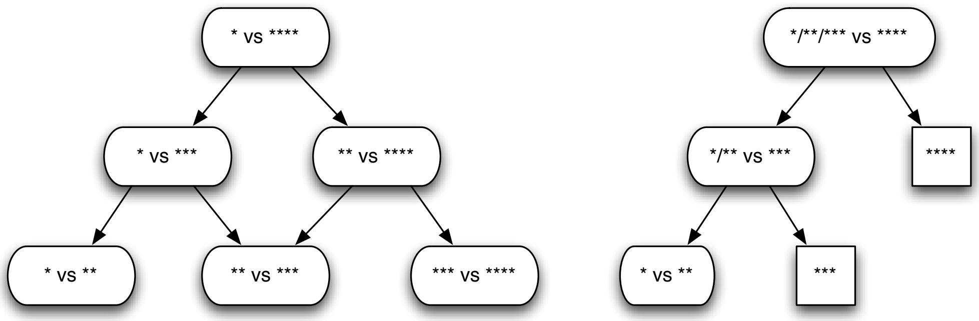


Figure 1: Top section of DAGSVM (left) versus MCST-SVM (right).

the DAGSVM and MCST-SVM approaches for a four-class example.

The MCST is constructed using Kruskal's algorithm (1956), which works in polynomial time (Algorithm 1). This algorithm requires a measure of the similarity between every pair of classes, which is calculated using the distance between a *representative vector* for each class (Section 3.2). The MCST is iteratively built in a bottom-up fashion, beginning with all classes as singleton nodes. In each iteration, the algorithm constructs a node comprising the most similar sets of classes from two previously generated nodes. The similarity between two sets of classes is the shortest distance between the representative vectors of the classes in each set. For instance, the shortest distance between the sets of classes {*/**} and {***/****} is min{dist(*,***), dist(*,****), dist(**,***), dist(**,****)}. An SVM is then trained to discriminate between the children of the constructed nodes.

With respect to the example in Figure 1, the classes {*} and {**} are first found to be the most similar, thus forming a node which discriminates between these two classes. In the next iteration, the classes {**} and {***} are found to be the next most similar, producing a new node which discriminates between {*/**} and {***}. Since the most similar sets are considered lower in the tree, the sets closer to the root of the tree are progressively more dissimilar, until the root node discriminates between the two most dissimilar sets of classes.

Our approach resembles DAGSVMs in that the structure of the decision tree is important. However, unlike DAGSVMs, the MCST-SVM structure is inferred on the basis of similarity between the observed *features* of the data, which are known, rather than the *labels* of the classes, which we are trying to predict. We assume that classes with adjacent labels are similar in the feature space, but if this does not happen in the training data, the MCST-SVM will yield a structure that exploits inter-class similarity irrespective of class labels. Further, our reliance on features supports experimentation with different methods for calculating inter-class similarity (Section 3.2). An additional advantage of MCST-SVM classifiers over the other schemes is that MCST-SVM requires only $k-1$ decision nodes for a k-class problem (and a maximum of $k-1$ decisions to make a prediction). That is, only $k-1$ SVMs must be trained, thereby reducing training time.

3.2 Class Similarity Measures

As mentioned in Section 3.1, the construction of an MCST-SVM classifier requires the computation of a similarity measure between classes. The MCST-SVM method may use any measure of inter-class similarity during the tree construction stage, and many such methods exist (e.g., linear discriminant analysis to order a tree of classifiers (Li et al., 2007)). We elected to use class prototypes to calculate similarity since they have achieved good performance in previous MCST-SVM applications (Lorena and de Carvalho, 2005; Bickerstaffe et al., 2007), and are fast to compute over many documents with a large feature space.

Algorithm 1 Constructing the MCST-SVM

```
1: Let $V$ be a set of graph vertices, where each vertex $v_i \in V$ represents rating class $i$ and its available training samples. $\forall i$ compute $r_i$, the class representative for rating class $i$.
2: Let $E$ be a set of graph edges. $\forall i, j$ where $i \neq j$, compute $e_{i,j} \in E$, the distance between class representatives $r_i$ and $r_j$.
3: Sort the members of $E$ in ascending order.
4: $\forall i$, let $S_i = v_i$, and add $S_i$ as a singleton node to the MCST-SVM tree $T$.
5: Let $i = 0$ and $j = 0$ be counting variables.
6: while $i < |V| - 1$ do
7:     Select the $j$-th edge according to the ordering of inter-class distances.
8:     if the vertices of the edge are in disjoint sets $S_p$ and $S_q$ then
9:         Define $S_p$ as a positive class and $S_q$ as a negative class.
10:        Let $S_t = S_p \cup S_q$, and add a new node containing $S_t$ to $T$.
11:        Connect the left and right branches of the node containing $S_t$ to the nodes containing $S_p$ and $S_q$ respectively.
12:        Remove $S_p$ and $S_q$.
13:        $i = i + 1$.
14:    end if
15:    $j = j + 1$.
16: end while
17: Train a binary SVM for each non-leaf node of $T$.
18: Return the MCST-SVM tree $T$.
```

We first determine a representative feature vector for each class, and then calculate the distance between these representative vectors.

Determining a representative vector. Each review is represented as a vector of boolean attributes, where each attribute indicates the presence or absence of a word or punctuation mark in the text. We elect to use boolean attributes since they have been shown to be advantageous over term-frequency approaches for sentiment detection, particularly when SVMs are employed (Pang et al., 2002). We considered two ways of determining a representative vector: *centroid* and *sample selection*.

- **Centroid.** Given N boolean feature vectors $\boldsymbol{a}_i$ of length n, compute the centroid vector $\boldsymbol{m}$ with values

$$m_j = \frac{1}{N}\sum_{i=1}^{N} a_{i,j} \quad \text{for } j = 1, \ldots, n \; . \qquad (1)$$

  This measure produces a representative vector that contains the proportion of training samples for which each feature occurs.

- **Sample selection.** From the training samples of each class, select one sample which maximises the average *Tanimoto coefficient* (Tanimoto, 1957) with respect to all other samples in that class. The Tanimoto coefficient is an extension of cosine similarity which yields the Jaccard coefficient for boolean feature vectors. Given two boolean vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, the Tanimoto coefficient is defined as

$$d_t(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{\|\boldsymbol{a}\|^2 + \|\boldsymbol{b}\|^2 - \boldsymbol{a} \cdot \boldsymbol{b}} \; , \qquad (2)$$

  where larger values of d_t indicate a higher degree of similarity between boolean vectors. This measure chooses a representative vector which on average has the most "overlap" with all other vectors in the class. We use Tanimoto distance, rather than the classical cosine similarity measure, since we employ boolean valued features instead of term-frequency features.

Calculating distance between vectors. We propose two methods to perform this task: *Euclidean distance* and the *Tanimoto coefficient*.

- **Euclidean distance** is used when the vectors that represent a class are centroid vectors (real-valued).

- The **Tanimoto coefficient** is used when the representative vectors of a class are boolean valued. It is calculated using Equation 2.

3.3 Irrelevant Feature Culling

The MCST-SVM scheme provides a natural mechanism for reducing the dimensionality of feature vectors in order to address the overfitting problem. This is due to the fact that each internal decision node is trained using *only* the samples that belong to the classes relevant to this node. The reviews for these classes are likely to omit some of the words that appear in the reviews for classes that are relevant to other nodes, in particular in the lower layers of the tree. Consequently, an internal node can be trained using a *subset* of the features that occur in the entire training dataset. This subset contains only those features which are observed in the samples relevant to training the node in question.[2] Section 4.2 shows that when tested on "real world" datasets, this method can remove thousands of irrelevant features and improve classifier performance, while reducing memory requirements and training times.

4 Experiments and Results

In this section, we evaluate the MCST-SVM classifier described in Section 3. First, we systematically compare the performance of the different variants of this method: (1) with or without culling irrelevant features, and (2) using the centroid/Euclidean-distance combination or the Tanimoto coefficient to measure inter-class similarity. We then compare the best of these methods with Pang and Lee's (2005). Our results show that a combination of relatively small improvements can achieve a substantial boost in classifier performance, yielding significant improvements over Pang and Lee's results.

All our experiments are performed with 10-fold cross validation, and the results are assessed using classification accuracy.[3] "Significance" refers to statistical significance determined by a paired t-test, with $p < 0.05$.

4.1 Dataset

Our experiments were conducted on the *Sentiment Scale* dataset (v1.0),[4] which comprises four sub-corpora of 1770, 902, 1307 and 1027 movie reviews with an associated mapping to a three and four-star rating for each review.[5] Each sub-corpus is written by a different author (denoted Author A, B, C and D respectively), thus avoiding calibration error between individual authors and their ratings. Review texts are automatically filtered to leave only subjective sentences (motivated by the results described in (Pang and Lee, 2004)); the mean number of words per review in each subjective-filtered sub-corpus is 435, 374, 455 and 292 respectively.

4.2 MCST-SVM Variants

Table 1 summarizes the results for the four MCST-SVM variants (the results that are statistically significant compared to the centroid/no-culling option are boldfaced).

Feature culling. Our results show that feature culling produces some improvement in classifier accuracy for all the three-class and four-class datasets. The impact of feature culling is statistically significant for all the four-class datasets when coupled with the Tanimoto coefficient. However, such an effect was not observed for the centroid/Euclidean-distance measure. In the three-class datasets, the improvements from feature culling are marginal for Authors A, B and C, but statistically significant for Author D (4.61%), both when using the centroid/Euclidean-distance measure and the Tanimoto coefficient. We posit that feature culling affects Author D because it reduces the overfitting problem, which caused the initially poor performance of MCST-SVM without culling on this author's short review texts (the reviews by this author, with 292 words on average, are the shortest in the Sentiment Scale dataset by a large margin, Section 4.1). Despite this improvement, all the MCST-SVM variants (as well as Pang and Lee's methods) exhibit worse performance for Authors B and D, who have shorter reviews, than for Authors A and C.

The culling of irrelevant features also has the benefit of reducing node training times and facil-

[2]The root node always considers all classes and therefore considers all features across the whole training dataset.

[3]We also have results for mean absolute error (MAE), which confirm our classification accuracy results.

[4]`http://www.cs.cornell.edu/People/pabo/moviereview-data.`

[5]In principle, classifiers for the three- and four-class ratings of the Sentiment Scale dataset could be enumerated using optimal stacks of SVMs. However, we wish to directly compare our method with Pang and Lee's (2005). Higher-discrimination datasets (for which optimal stacks are infeasible) will be tested in the future.

	Centroid, no culling	**Tanimoto, no culling**	**Centroid, with culling**	**Tanimoto, with culling**
Three-class				
Author A	70.396	70.396	71.017	71.997
Author B	60.556	60.556	61.111	61.111
Author C	75.154	75.481	76.231	76.923
Author D	59.608	59.608	**64.216**	**64.216**
Four-class				
Author A	62.429	63.810	63.090	**65.720**
Author B	49.111	49.792	50.622	**52.890**
Author C	64.846	65.689	65.692	**66.985**
Author D	49.118	49.626	51.177	**51.873**

Table 1: Performance accuracy (percentage correct predictions) for MCST-SVM variants.

itating a memory-efficient implementation. For example, without feature culling, the nodes of an MCST-SVM for Author A in the four-class dataset take training samples with 19752 features. In contrast, when irrelevant feature culling is applied, the number of features for each of the two non-root decision nodes reduces to 15445 and 17297. This corresponds to a total space saving of 6582 features $((19752 - 15445) + (19752 - 17297))$, yielding an in-memory reduction of 16.7%. Such memory reductions are particularly important for large datasets that may have trouble fitting within typical memory limitations. Node training times are also reduced by up to approximately 10%.

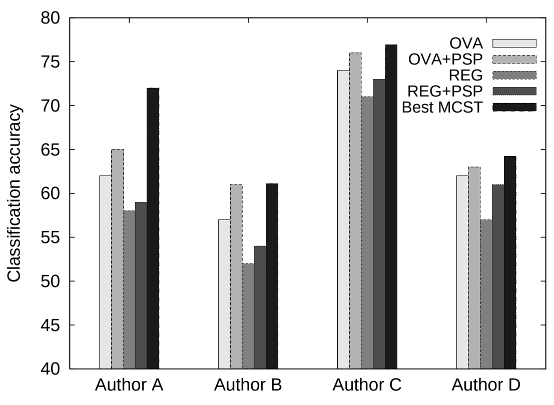
Class similarity measures. As mentioned above, Table 1 shows that the Tanimoto coefficient, coupled with feature culling, yields marginally better results than the centroid/no-culling option for most authors in the three-class dataset, and significantly better results for all the authors in the four-class dataset. The Tanimoto coefficient generally matches or outperforms the centroid/Euclidean-distance measure both with feature culling (Columns 4 and 5 in Table 1) and without feature culling (Columns 2 and 3). However, without feature culling, these improvements are not statistically significant.

For most cases in the three-star dataset, the tree structures found using the Tanimoto coefficient are identical to those found using the Euclidean-centroid option, hence the performance of the classifier is unchanged. For some validation folds, the Tanimoto coefficient discovered tree structures that differed from those found by the Euclidean-centroid option, generally yielding small accuracy improvements (e.g., 0.98% for Author A in the three-star dataset, with feature culling). The Tanimoto coefficient provides a greater benefit for the four-class dataset. Specifically, when feature culling is used (Columns 4 and 5 in Table 1), accuracy improves by 2.63% and 2.27% for Authors A and B respectively (statistically significant), and by 1.29% and 0.70% for Authors C and D respectively. This may be explained by the fact that there are many more tree structures possible for the four-class case than the three-class case, thereby increasing the impact of the inter-class similarity measure for the four-class case. However, this impact is significant only in conjunction with feature culling.

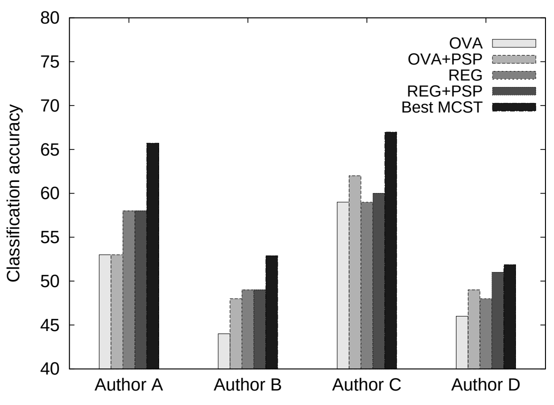
4.3 Comparison with Pang and Lee (2005)

Figure 2 compares the performance of the algorithms presented in (Pang and Lee, 2005) against the performance of the best MCST-SVM variant, which employs feature culling and uses the Tanimoto coefficient to compute inter-class similarity (Section 4.2). As per (Pang and Lee, 2005), REG indicates SVM-R, which is the baseline ordinal regression method. The suffix "+PSP" denotes methods that use the metric labeling scheme. We excluded DAGSVM from our results to maintain consistency with Pang and Lee's experiments. However, according to (Platt et al., 2000), the performance difference between DAGSVM and OVA is not statistically significant.

Generally, the MCST-SVM is competitive against all the classifiers presented in (Pang and Lee, 2005), and in some cases significantly outperforms these methods. Specifically, the hierar-



(a) Three-class data.



(b) Four-class data.

Figure 2: Best MCST-SVM versus competing methods.

chical classifier outperforms OVA+PSP by 7% in the three-class case for Author A (statistically significant), while in the four-class case the MCST-SVM outperforms the best competing methods by 7.72%, 3.89% and 4.98% for Authors A, B, and C respectively (statistically significant). The small improvement of 0.87% for Author D indicates that our approach has the most impact for reviews that contain a relatively large amount of subjective text.

5 Conclusion and Future Work

This paper described a hierarchical classifier applied to multi-way sentiment detection. The classifier is built by exploiting inter-class similarities to arrange high-performance binary discriminators (SVMs) into a tree structure. Since our inter-class similarity measures are based on sample features, they make the problem of structure determination tractable, and enable experimentation with different similarity measures. The resultant structures provide a natural mechanism to remove irrelevant features at each level of the tree, thus reducing the dimensionality of the feature space, which in turn reduces memory requirements. Importantly, these benefits are achieved while improving upon state-of-the-art classification performance, in particular with respect to higher-discrimination datasets.

The MCST-SVM classifier can be generalised to any number of classes, and is extendable in the sense that the classifier algorithm employed in each tree node may be replaced by other classifier algorithms as technology advances. The MCST-SVM classifier is also versatile, and may be applied to variations on the rating classification problem, e.g., traditional text classification.

The MCST-SVM algorithm is not specific to sentiment detection. However, it has several properties which make it particularly suitable for the rating inference problem. Firstly, the MCST-SVM accounts for inter-class similarity and is therefore capable of capturing the ordinal nature of ratings. Secondly, the tree structures permit irrelevant feature culling, which in turn reduces memory requirements and training times.

Future work will involve testing our approach with higher-discrimination datasets, developing methods to pre-process review texts (e.g., improved negation tagging, and incorporating part-of-speech tagging), and further addressing the problem of overfitting. To this effect we will investigate different feature selection algorithms, e.g., (Weston et al., 2003), and their utilisation within the classifier trees. We also propose to consider aspects of reviews (Snyder and Barzilay, 2007), and investigate other methods that measure class similarity, such as selecting typical instances (Zhang, 1992).

Acknowledgments

This research is supported in part by ARC grant LP0883416 and GapBuster Worldwide.

References

Allison, B. 2008. Sentiment detection using lexically-based classifiers. In *Proceedings of the 11th International Conference on Text, Speech and Dialogue*, pages 21–28, Brno, Czech Republic.

Beygelzimer, A., J. Langford, and P. Ravikumar. 2009. Error-correcting tournaments. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, pages 247–262, Porto, Portugal.

Bickerstaffe, A., A. Lane, B. Meyer, and K. Marriott. 2007. Building smart diagram environments with domain-specific gesture recognizers. In *Proceedings of the 7th IAPR International Workshop on Graphics Recognition*, pages 145–156, Curitiba, Brazil.

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

Goldberg, A.B. and X. Zhu. 2006. Seeing stars when there aren't many stars: Graph-based semi-supervised learning for sentiment categorization. In *TextGraphs: Workshop on Graph Based Methods For NLP*, pages 45–52, New York, New York.

Hsu, C. W. and C. J. Lin. 2002. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425.

Koppel, M. and J. Schler. 2006. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109.

Kruskal, J. B. 1956. On the shortest spanning subtree and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50.

Li, T., S. Zhu, and M. Ogihara. 2007. Hierarchical document classification using automatically generated hierarchy. *Journal of Intelligent Information Systems*, 29(2):211–230.

Lorena, A. C. and A. C. P. L. F. de Carvalho. 2005. Minimum spanning trees in hierarchical multiclass Support Vector Machines generation. *Innovations in Applied Artificial Intelligence*, 3533:422–431.

Mao, Y. and G. Lebanon. 2006. Isotonic conditional random fields and local sentiment flow. In *Proceedings of the 20th Annual Conference on NIPS*, pages 961–968, British Columbia, Canada.

McDonald, R., K. Hannan, T. Neylon, M. Wells, and J. Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the 45th Annual Meeting of the ACL*, pages 432–439, Prague, Czech Republic.

Pang, B. and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the ACL*, pages 271–278, Barcelona, Spain.

Pang, B. and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 115–124, Ann Arbor, Michigan.

Pang, B., L. Lee, and S. Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 79–86, Philadelphia, Pennsylvania.

Platt, J. C., N. Cristinini, and J. Shawe-Taylor. 2000. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12:547–553.

Smola, A. and B. Schölkopf. 1998. A Tutorial on Support Vector regression. Technical Report COLT NC-TR-98-030, University of London.

Snyder, B. and R. Barzilay. 2007. Multiple aspect ranking using the Good Grief algorithm. In *Proceedings of HLT/NAACL*, pages 300–307, Rochester, New York.

Tanimoto, T.T. 1957. IBM internal report.

Weston, J., A. Elisseff, B. Schölkopf, and M. Tipping. 2003. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3:1439–1461.

Wilson, T., J. Wiebe, and P. Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 347–354, Vancouver, Canada.

Yu, H. and V. Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the Conference on Empirical Methods in NLP*, pages 129–136, Sapporo, Japan.

Zhang, J. 1992. Selecting typical instances in instance-based learning. In *Proceedings of the 9th International Workshop on Machine Learning*, pages 470–479, Aberdeen, Scotland.