Refined Experts

Improving Classification in Large Taxonomies

Paul N. Bennett
Microsoft Research
One Microsoft Way
Redmond, WA
paul.n.bennett@microsoft.com

Nam Nguyen[*]
Cornell University
Department of Computer Science
Ithaca, NY
nhnguyen@cs.cornell.edu

ABSTRACT

While large-scale taxonomies – especially for web pages – have been in existence for some time, approaches to automatically classify documents into these taxonomies have met with limited success compared to the more general progress made in text classification. We argue that this stems from three causes: increasing sparsity of training data at deeper nodes in the taxonomy, error propagation where a mistake made high in the hierarchy cannot be recovered, and increasingly complex decision surfaces in higher nodes in the hierarchy. While prior research has focused on the first problem, we introduce methods that target the latter two problems – first by biasing the training distribution to reduce error propagation and second by propagating up "first-guess" expert information in a bottom-up manner before making a refined top down choice. Finally, we present an empirical study demonstrating that the suggested changes lead to 10-30% improvements in F1 scores versus an accepted competitive baseline, hierarchical SVMs.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; I.5.4 [**Pattern Recognition**]: Applications - Text processing

General Terms

Algorithms

Keywords

Text Classification, Large-scale Hierarchy

1. INTRODUCTION

Web taxonomies (*e.g.* ODP, Yahoo!) organize the content of the web into deep hierarchies containing hundreds of thousands of categories. The usefulness of taxonomies to improve browsing and other search activities has been demonstrated repeatedly [7, 8, 26]. However, the actual number of web documents (billions) far exceeds the number that have been manually placed into the taxonomies (millions). This combined with the fast-growing pace of the web as well as dynamically generated web-pages argues for the need for hierarchical classification methods that can automatically place web pages into a taxonomy.

[*]This work was done at Microsoft Research.

To this end, a variety of researchers have developed and studied methods for hierarchical classification [2, 3, 5, 6, 8, 9, 12, 15, 17, 22, 25]. However as noted by Liu *et al.* [15] the general performance achieved in hierarchical classification, especially at lower levels of the hierarchy, has been far lower than what has been achieved in many isolated realms of text classification despite attempts to leverage the information provided by the hierarchy.

Researchers often attribute a large part of the blame for poor performance on the sparsity of labeled training data at lower nodes in the hierarchy, and methods aimed at addressing this, such as shrinkage [16], have met with varying degrees of success. However, we argue that *data sparsity* is simply one of the challenges that hierarchical classification presents. In particular, hierarchical classification also faces two other signification challenges: *error propagation* and *non-linear decision surfaces*.

Error propagation occurs when a classification mistake made higher in the hierarchy causes errors lower down. This in turn leads to the more subtle effect that the test distribution drifts from the training distribution as we move lower in the hierarchy. Classification at lower levels is done assuming only documents that actually belong in a subhierarchy rooted at a node will be classified at that node; when in reality, documents misclassified at higher levels must also be classified. Basic machine learning theory informs us that a more optimal decision surface could be learned if we trained on the same type of distribution as will be seen during testing. By predicting the behavior of the classifier, we can do just that, and we introduce a method called *Refinement* that takes advantage of this to significantly improve classification performance of a hierarchical SVM.

In addition, non-linear decision surfaces present a problem for hierarchical classifiers where the primary text classification models are linear. The problem of non-linearity has been relatively unrecognized in the hierarchical classification community, but it is well-known in the theoretical machine learning community that a union of linear surfaces is a non-linear surface [11] (*cf.* Figures 3 and 4). Starting from the assumption that given enough training data a linear classifier would work well at the leaf nodes, then, in general, we must expect non-linear surfaces to arise at the interior nodes since they are a union of their children. While one solution is to simply introduce general non-linear classifiers, this could also lead to overfitting by introducing a more general model than necessary. However, given knowledge of which lower-level categories a documents belongs to, we could introduce just the right amount of non-linearity by including the membership in lower-level categories as a feature. Although the actual lower-level membership is not available when classifying, based on this intuition, we introduce a general method we call *Refined Experts* that takes a rough guess at the lower-level category classification and propagates this information up through the hierarchy before taking a top-down classification pass that refines the bottom-up guesses. Empirical results demonstrate that this further improves the gains achieved by Refinement, yielding a method that has 10-30% performance improvements over a hierarchical SVM.

2. RELATED WORK

While a large body of work exists on hierarchical classification, here we simply highlight some of the more relevant work. McCallum *et al.* [16] were the first to deal with data-sparsity at the lower-level categories of a hierarchy by applying *shrinkage*, an established statistical technique that smooths the parameter estimates of a child with its parent in order to obtain more robust parameter estimates. Although we do not directly address data sparsity, shrinkage is an orthogonal complementary improvement that could be used in addition to our methods.

Jordan and Jacobs [10] introduced Hierarchical Mixtures of Experts (HME), which is a tree structure comprising expert nodes at the leaves and gating nodes at the internal nodes. HME is formulated as a maximum likelihood estimation problem which can be solved via an EM algorithm. In addition, Ruiz & Srinivasan [20] developed a hierarchical neural network model using a variation of HME where both the expert nodes and gating nodes are implemented as neural networks. In both HME and Refined Experts, the learned models for each internal category takes into account the outputs of children categories. However, in our method, categories at the same level can be trained in parallel – an important difference when applying a method to large-scale taxonomies where joint optimization may not be feasible. In addition, in our method, the base classifier can be replaced by any other available learner. We also introduce and analyze changes in the training distribution at the nodes which creates a very different effect than the gating of HME.

Liu et al. [15] evaluated performance of Hierarchical Support Vector Machines [8] in web-page classification over the hierarchy of the Yahoo! categories. The authors also pointed out the difficulties in applying text categorization algorithms to large-scale Web taxonomies and the need for improvement in large-scale hierarchical categorization. Our work complements theirs by showing improvements over the baseline they found to be best (Hierarchical SVMs).

Finally, Xue *et al.* [24] proposed a two stage approach to large-scale hierarchical classification. In the first stage, the search stage, a category-search algorithm is used to obtain category candidates for each document – essentially creating a dynamic smaller hierarchy to classify into. Then in the second stage, the classification stage, the classification model is trained on this small subset of the original hierarchy. Their approach reduces error propagation by considering fewer categories and thus reduces the likelihood of an error. They also augment this with an approach related to shrinkage that collapses the data with data from ancestor categories. Our approach stems from a similar motivation; however, we target the underlying machine learning problems. As a result, it is possible to combine the two approaches, using their approach to restrict the set of nodes considered and ours to do the classification. Furthermore, they report only microF1, which is dominated by common categories, but presumably if one's goal is to classify into a large number of categories, the target is to be accurate over all those categories, which is better captured by macroF1. In our empirical section, we demonstrate through the performance of one method that, because of the power law fall-off of category size, it is possible to significantly improve microF1 while hurting macroF1 (see Figure 7).

3. PROBLEM APPROACH

In contrast to previous work, we identify and address the underlying machine learning issues behind two key problems of hierarchical classification: error propagation and complex (non-linear) decision surfaces. To this end, we enrich the understanding of the fundamental challenges hierarchical classification presents. From a technical point of view, our approach has much in common with HME, however, it is applicable both to a wide-range of base classifiers and can be used over large-scale taxonomies where joint optimization may not be computationally feasible. Likewise, while the motivation is related to [24], we target improvements in *macro*F1 and develop methods that can be used in conjunction with their methods in addition to the flexibility of plugging in other base classifiers. Finally, we include a series of ablation studies that help both to verify important modeling decisions in our approach and highlight important properties of hierarchical classification more generally.

3.1 Error Propagation and Refinement

Standard approaches to hierarchical classification build a classifier at each node in the hierarchy. Each classifier is typically trained in one of two ways. In the *flattened approach* the training data are the documents belonging to the class, positive data, versus all documents not belonging to the class, negative data. In the *hierarchical approach* the training data are the documents belonging to the class, positive data, versus those documents belonging to the parent but not to the class, negative data. The hierarchical approach has been shown to have superior performance in previous work [15], and thus we restrict ourselves to this. In order to classify the data, the classifiers are run in a top-down fashion. That is, a Pachinko model [14, 17] is employed – first classifying the document at the uppermost level and then for each classifier that "fires" (predicts positive), classifying the document at the next lower level. Since topics are not disjoint, documents can belong to multiple children and one common approach is to build a binary classifier per topic; we take this approach here.

In practice, as one progresses down the hierarchy during classification, the actual distribution of examples a classifier is predicting over changes from that used during training in two ways due to errors at the higher levels. First, docu-

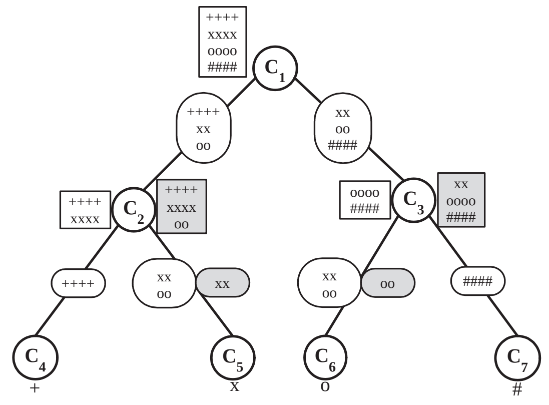

Figure 1: Refinement uses predicted behavior to change the training distribution and learns models that filter out errors. The *training* set for the standard approach is illustrated with white boxes and in gray boxes when Refinement would differ. The effect on a *test* set is illustrated with standard classifications in white ovals and in gray when Refinement would differ.

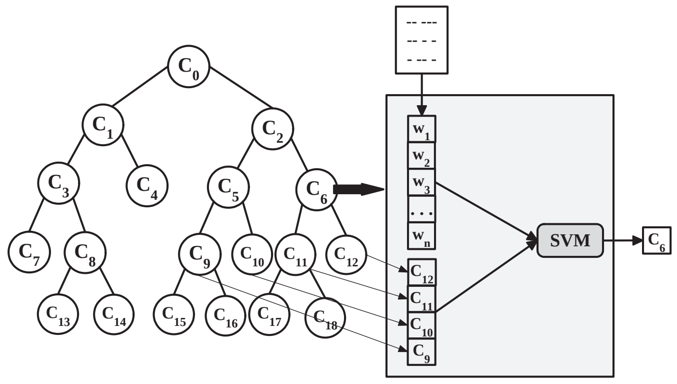

Figure 2: Refined Experts augments the representation of a document with a set of metafeatures containing membership predictions from lower nodes (here its children and their "cousins").

ments that should have come down this branch have been excluded (false negatives), and second, documents that should not have come down this branch have been included (false positives). Correcting the former requires a stronger signal to get the document down this (the correct) branch; this is addressed in the next section. For false positives, we would ideally identify these errors early and not pass them on,[1] impacting classes downstream as little as possible.

This problem can be addressed by an approach we call *Refinement* since it filters errors out. The key is a simple application of cross-validation. In particular, we perform cross-validation over the training data and use the predicted labels to filter the training data to a node – thus aligning the training distribution with what will likely occur during testing. Then the model is trained using the actual labels as usual. We do this with one slight modification. As is well known, in text classification, the number of positive documents belonging to a class is quite small on average, and classifier accuracy often crucially depends on the number of positive documents available and not simply the total number of documents. If we used the predicted distribution, we would lose some of the positive documents because of false negatives. Because of the scarcity of positive data, we hypothesize this would have a negative impact on the classifier accuracy. We thus modify this approach, and as training data at a node use the union of the predicted distribution and the actual distribution, *i.e.* the training data standardly used unioned with the predicted errors.

This is illustrated for an example hierarchy in Figure 1 where instances from C_5 and C_6 are confused with each other half the time. The figure shows a case where the training and test set have 16 examples whose membership is balanced between the leaf nodes. Because Refinement sees instances similar to error instances when training, it can learn models that halt propagation – C_6's "o" instances at node C_2 and C_5's "x's" at C_3 – thus improving accuracy at lower levels.

Because the predicted distribution at a node depends on the predictions above the node, the classifiers have a training dependency that requires all ancestors of the current node to be trained. Since the number of nodes in a tree grows exponentially, this dependency does not significantly reduce the amount of parallelization that can be utilized during training. For example, in a full binary tree, if we assume training proceeds by level, the number of nodes to be trained at the current level would be one greater than the number of total nodes trained so far. Thus, when doing parallel training – which approaches to large-scale taxonomies typically rely on – the growth ensures that the available number of processors are quickly saturated. The primary computational cost of this approach is in the n-fold cross-validation which introduces a linear cost of n times the standard approach.

3.2 Nonlinearity and Expert Information

A second problem faced in hierarchical classification is that the upper levels of a hierarchy often contain general concepts (*e.g. Kids & Teens*) that are hard to discriminate because they cover a very diffuse set of topics. We can characterize this more precisely as a non-linear decision surface and illustrate this through an example. Consider separating the concepts in the hierarchy in Figure 3 where our training and test data are drawn from multivariate Gaussians resulting in a distribution illustrated in Figure 4.

Note that given the hierarchy of Figure 3, a standard hierarchical classifier would first attempt to separate the "x" class from the remaining classes. Even though the "x" class can be separated from the remaining data quite well, any *linear* decision surface cannot separate it accurately.

However, given the outputs from a linear decision surface trained to separate the circles from the remainder and one separating the plusses from the remainder, it is quite easy to learn a non-linear surface (shaped like a rotated "Z" here) with near minimal error at the root. While it is possible to create an alternate example where the union of two classes results in a linear decision surface, in general, the union of linear surfaces results in a non-linear surface [11]. Although we could use a general non-linear learner, it seems intuitive instead to restrict the type of complexity we introduce. In

[1]The ultimate goal would be to identify errors and make a new attempt to classify. However, this is "risky" in case the document is not an error. Simply stopping the document balances the benefit of not passing on an error with the risk of misidentifying a document as an error.

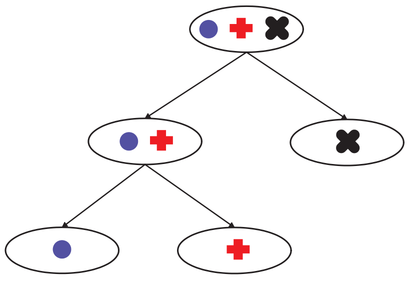

Figure 3: A hypothetical hierarchical problem consisting of the "circle", "plus" and "x" classes.

particular, since the resulting problem is linear given the inputs from the children, we can use just such a model.

In order to do this, we adapt an approach from metaclassification and the combination of classifiers [1]. We introduce predictions from the lower nodes as metafeatures at the higher levels. We do this by first training linear classifiers at the leaf nodes using cross-validation over the training data and then use the predictions over the *training data* gathered during cross-validation as metafeatures available at the next higher level. This is illustrated in Figure 2.

Putting this together with Refinement, this suggests a bottom-up training pass which helps drive examples down the correct branch, correcting false negatives, followed by a top-down training pass that halts the propagation of false positives down an incorrect branch. We call this *Refined Experts* since the bottom-up pass can be seen as passing up guesses from specialized classifiers,[2] *i.e.* experts, followed by a Refinement pass using the enriched representation.

A subtle point is that no filtering should occur in the bottom-up pass like that in top-down. That is, a bottom-up classifier does not require one of the classifiers below it to fire in order to fire. This would be undesirable since the leaf nodes have the least training data and may not fire often. Thus, at interior nodes, the model can learn to pass a signal from lower-down up, or using the document features, directly detect a document is a member – increasingly likely as more training data becomes available at higher nodes.

We also must choose how much bottom-up information to use. To restrict the model to only the most pertinent information, we use predictions from a node's children and their cousins (see Figure 2). The cousins are included since a high probability at a cousin node implies we may want to downweight information from the children since the document likely belongs to a sibling.

For the bottom-up pass, one detail remains to be decided. What subset of the training data should be used for training at each node? Since no filtering is occurring, one natural choice might be a flattened classifier but with access to the metafeatures, *i.e.* the positive data are all documents belonging to the node (or its descendants) and the negative data are all documents not belonging to the node (or its descendants). Note that there is no reason during the bottom-up pass to learn to discriminate a node from its siblings since the point is to generate enough signal for the

[2]In a sanity check for the method, it is indeed the case that if the predictions are completely accurate for the bottom-up classifiers at the leaf node, the resulting full classifier attains near perfect performance.

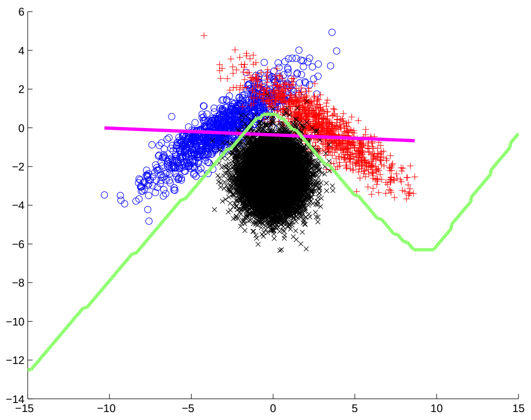

Figure 4: Data for the Hypothetical Hierarchical Task in Figure 3. When trained on holdout data, the linear surface that is learned at the root node using the standard approach (*magenta*) versus what would be learned when the input space is extended to use predictions from the circle and plus classifier (*green*).

document to make it to the parent during a top-down pass; it will then be the job of the top-level classifier to make the distinction between siblings. We therefore use all documents belonging to the node (or its descendants) as positive data and all documents not belonging to the node's *parent* (or its descendants) as negative data.

In Refined Experts, while both the bottom-up and top-down models built at a given node use the predictions from the nodes below it, they may learn different weights for those predictions. This reflects differences in their training sets distributions and what set of documents they are trying to discriminate the given class's documents from.

4. EXPERIMENTAL ANALYSIS

4.1 Data

In order to empirically investigate our methods, we have used the Open Directory Project's (ODP) hierarchy of the web [18]. Using a list of urls and categories downloaded in Dec 2007, we crawled and cached the content in Jan 2008. This resulted in about 4.2M web pages in 600K categories.

Since they mainly contain foreign pages or geographic distinctions, we chose not to examine the *World* and *Regional* branches. This left 1.7M documents in 173K categories.

We chose to focus on classification using the content of the web pages only. Integrating more information (*e.g.* the urls) is simply likely to improve both the baseline and our methods. The web pages were tokenized retaining all tokens occurring in at least 10 documents (421711 tokens) and randomly split (70/30) into a train/test set with about 1.2M/509K examples. Most documents have at least three levels assigned to them. Most categories are at level 5 or 6.

Because our focus is on macroF1, we only keep categories with at least one document in both the train and test set. This ensures better statistical stability of the F1 measure while leaving a very large number of classes as well as deep hierarchies. This reduces the number of categories to 62,767.

To explore comparisons between various methods, it was computationally pragmatic to limit the number of categories further. We selected a number of sub-hierarchies to perform prediction over as if they were individual hierarchies and then for overall performance we combine their performance measures together weighted by the size of the hierarchy (adding classes for macro and contingency tables for micro). We also give the straight arithmetic average across the hierarchies, giving equal weight to each, which has close to the same value as the weighted version. We denote these as *Overall* and *Average* respectively. We selected 9 of the 15 top-level sub-hierarchies to cover a variety of topics. These sub-hierarchies include: Adult, Computer, Game, Health, Home, Kids & Teens, Reference, Science, and Shopping.

4.2 Models

Baseline - Hierarchical SVM

Since SVMs have shown success in text categorization [9, 15], we use a Linear-SVM as the base classifier in our refined experts model as well as in the baseline hierarchical SVM. We used stochastic gradient descent [21] for our implementation of SVM. For comparability to other published results, we set C using a data-driven heuristic based on the radius containing the training data as done in SVM-Light [9], $C = \sqrt{\sum_{i=1}^{n} \frac{\|x_i\|^2}{n}}$. Because a document can belong to multiple categories, we build a binary classifier for each node in the hierarchy where the positive data consists of the training data labeled as belonging to the class (or a descendant of the class) and the negative data is all data belonging to that node's parents that does not belong to the node. As is standard in hierarchical SVMs, when a document is classified as a member of a class, classification continues into the subhierarchy beneath the class's node until no more positive predictions are made. To make the baseline as competitive as possible, we tune the thresholds of each classifier by performing 5-fold cross-validation and choosing a threshold which optimizes F1 (*i.e.* s-cut optimization [15]).

Refinement

The model for Refinement training is the same as for the hierarchical SVM except that, as discussed in Section 3.1, the training distribution is the union of the documents at the node's parent with the documents predicted to reach the node's parent. The predictions over the training set are gathered with 5-fold cross-validation on the training set.

Refined Experts

For the Refined Experts model described in Section 3.2, the bottom-up pass uses the feature set augmented by metafeatures that contain predictions over the training data (again gathered via cross-validation) from the bottom-up models of the node's children and their cousins. The top-down pass is the same as in the *Refinement* model. However, the feature set is now augmented by metafeatures containing the bottom-up model predictions. These predictions are the probability output by a calibrated sigmoid [19].

4.3 Performance Measures

To compare the performance of the classification methods we look at a set of standard performance measures. The F1 measure [23] is the harmonic mean of precision and recall where $Precision = \frac{True\ Positives}{Predicted\ Positives}$ and $Recall = \frac{True\ Positives}{Actual\ Positives}$. As is usual, we look at their *macro* average which weights the F1 for each class equally and *micro* average which weights each binary prediction equally [25]. We remind readers that since most classes are rare in text classification, macro-averaging tends to emphasize rare classes while micro-averaging tends to emphasize common classes.

While it is possible to use alternative performance measures (*e.g.* graph distance [6]) as a performance measure, we prefer to stay with macro and micro F1 which are more standard for text classification and well-understood. However, given the flexibility of our approach, it is simple to plug in a base learner that optimizes a different underlying metric.

4.4 Experimental Methodology

Other than the metafeatures, the representation is held constant for each of the methods. The portion of the document vector corresponding to the words is represented using a tfidf representation, retaining all features, and normalized to the unit sphere. The metafeatures are normalized separately by the sum of their absolute values. In the face of no feature selection, we rely on regularization (L2-norm) to control for noise in irrelevant features which empirical evidence shows works well in practice [13].

Since the underlying optimization algorithm uses stochastic gradient descent, each reported measure is actually the average of 10 runs using different random seeds.

Because our primary focus is on the performance across all classes, we compare macroF1 scores with a two-sided macro t-test for statistical significance [25].

4.5 Results

The results are summarized in Table 1. Differences in the macroF1 of all three methods are statistically significant at the p=0.05 level according to a two-sided macro t-test. Using Bonferroni correction, only Refinement vs. Refined Experts for *Adult* loses significance. Figure 5 presents the results broken down by hierarchy level.[3] Finally, Figure 6 gives the improvement relative to the baseline.

	Baseline		Refinement		Refined Experts	
	Macro	Micro	Macro	Micro	Macro	Micro
Adult	0.167	0.390	0.171	0.421	**0.181**	**0.440**
Computer	0.228	0.252	0.233	0.278	**0.270**	**0.321**
Game	0.376	0.417	0.430	0.486	**0.468**	**0.561**
Health	0.341	0.443	0.373	0.509	**0.401**	**0.540**
Home	0.372	0.430	0.397	0.459	**0.405**	**0.510**
Kids & Teens	0.288	0.385	0.302	0.451	**0.324**	**0.514**
Reference	0.351	0.571	0.378	0.631	**0.436**	**0.688**
Science	0.300	0.350	0.315	0.396	**0.355**	**0.485**
Shopping	0.332	0.309	0.366	0.368	**0.421**	**0.439**
Average	0.306	0.394	0.329 (7.6%)	0.444 (12.7%)	**0.362 (18.4%)**	**0.500 (26.8%)**
Overall	0.302	0.365	0.326 (7.9%)	0.414 (13.2%)	**0.361 (19.6%)**	**0.468 (28.0%)**

Table 1: Refinement outperforms the Baseline on every subhierarchy, and Refined Experts outperforms both.

4.6 Discussion

First, we note that the general trend in macro and micro F1 for the baseline as well as their absolute values are consistent with the literature [15]. Thus, the baseline is com-

[3]Results start at Level 2 since these are subhierarchies.

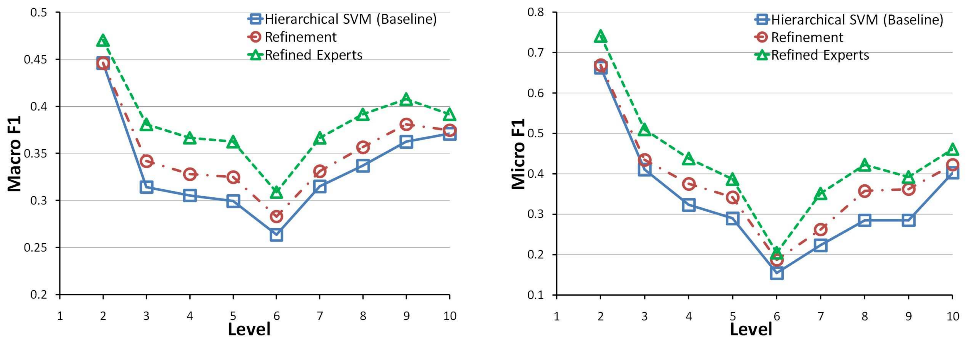

Figure 5: Macro (*left*) and Micro (*right*) F1 broken down by level of the hierarchy.

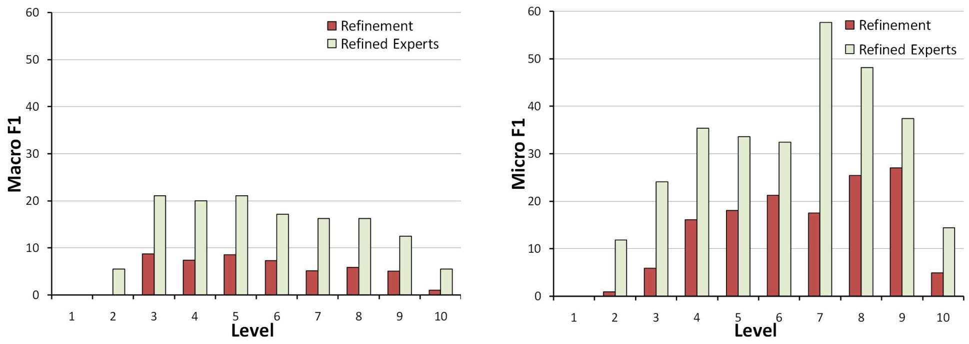

Figure 6: Improvement relative to baseline for Macro (*left*) and Micro (*right*) F1 broken down by level of the hierarchy.

petitive.[4] The dip that takes place around level 6 seems to be primarily attributable to categories in the hierarchy that are non-topical as well as leaf nodes at this level with few training examples. An example of the former is an organization of the documents in a branch alphabetically by country or language, *e.g.* Country/[A-Z]/Australia, Argentina, ... Thus, even though it's possible to classify the higher-level and lower-level categories with word based features, classifying into the alpha categories without character based features is difficult. Rather than focusing on optimizing for this hierarchy, we focus on techniques that work for any hierarchy since there are a variety of structures like this that might prove more challenging at the middle levels (*e.g.* by genre, author, *etc.*).

In comparing the methods, we see that Refinement improves over the baseline (7.9% macro and 13.2% micro) significantly, although not near the upper-levels of the hierarchy. This lack of improvement at the upper level is expected since Refinement simply modifies the baseline by halting the effects of error propagation. At the top, it is the same as the baseline, and it achieves greater gains as we progress down the hierarchy. Given the computational cost of prediction – the most important computational factor when attempting to classify the entire web – is exactly the same for Refinement as the baseline, this provides an easy alternative.

When examining the results for Refined Experts, we see it yields significant gains (19.6% macro and 28.0% micro) relative to both the baseline and simple Refinement. Furthermore, while Refinement cannot introduce any gains at the top-level, we see the full Refined Experts approach does by propagating up the expert metafeatures. Although obtaining this bottom-up information bears a computational cost, it is possible in future work computational accuracy can be balanced with the likelihood of improvement by taking the learned weight on a lower node's metafeature into account and short-circuiting the bottom-up evaluation.

4.6.1 Ablation Experiments

In order to investigate several of the algorithmic design choices we made, we conducted ablation experiments to examine performance relative to the methods above. The first two ablation experiments are over the full sets of subhierarchies we examined. The last two are smaller scale and were run over a (different) single subhierarchy chosen randomly.

First, we were interested in the choice of the data distribution used to define the training set for Refinement. Figure 7 compares the baseline, standard Refinement, and prediction-based Refinement (using just the predicted labels to define the training set). We see that prediction-based Refinement improves microF1 but actually produces the worst macroF1. This can be understood by recalling that most categories are rare (have few positive examples) and the percentage of categories that are rare increases as one goes down the hierarchy. Thus, using the predicted labels yields a gain primarily for the common classes but discards too many positive examples for rare categories. This also suggests a heuristic or function that weights between the two approaches based on the number of positive examples is a promising future area.

Next, we were interested in tradeoffs in how much bottom-up information to propagate in constructing the metafeature

[4]At the deepest level of the hierarchy, there is high variance from the small number of categories.

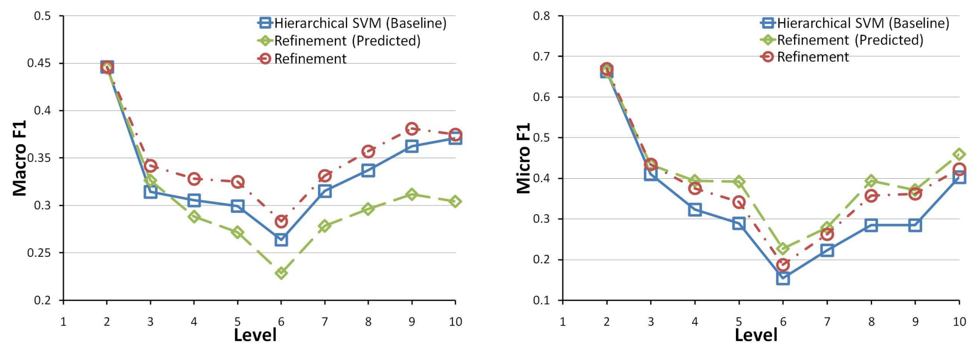

Figure 7: Impact of Training Distribution Choice on Refinement

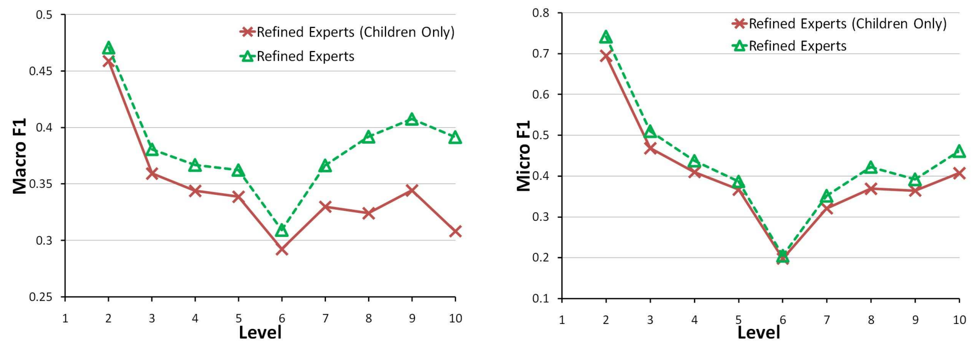

Figure 8: Impact of Choice of Metafeature Set

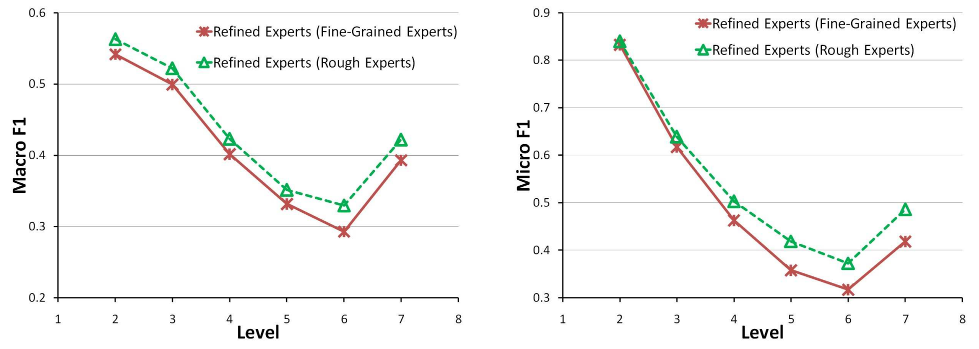

Figure 9: Impact of Choice of Expert Training Set/Discrimination Task

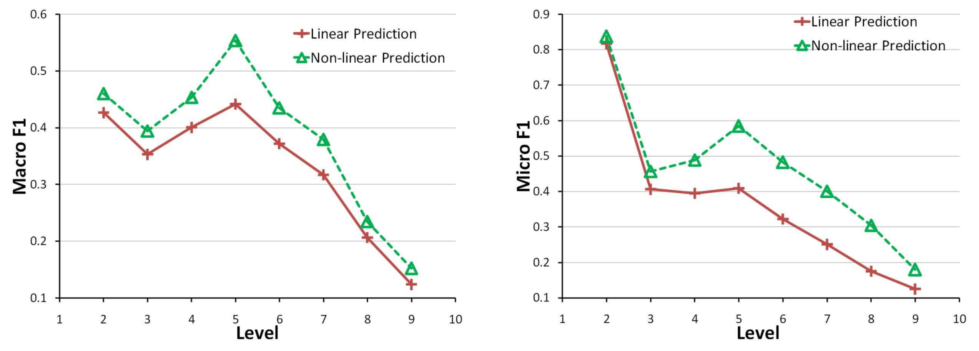

Figure 10: Importance of Nonlinear Transform

set. Figure 8 shows our standard Refined Experts which uses predictions from all children nodes plus their cousins versus Refined Experts using predictions from just the child nodes. We see that the cousins do in fact provide information, especially at the lower-level where information can be sparse.

Then, we did a smaller ablation study on the effects of the choice in how the bottom-up experts are trained (Figure 9). In the standard approach, we try to predict the examples at the node (positive) versus those not belonging to the node's parent or its descendants (negative) under the intuition that it's okay to make a false positive on a sibling category in the bottom-up process since the top-down prediction can fix that. The alternative displayed in the figure is for building "fine-grained experts" that try to discriminate documents belonging to the node (positive) from those not belonging to the node. We see that the "rough" first-guess is superior, primarily because it gives enough bottom-up signal to get an example headed down the correct branch. This gain is fairly uniform across the levels.

Finally, we conducted an experiment to investigate how important it was to use a non-linear transform (in our case a calibrated sigmoid) for the bottom-up expert predictions or if we could simply use the margin score from the SVM. This is in some way a test of how much nonlinearity plays a role in hierarchical classification – the reason being that when the expert prediction is the margin score, the model can be rewritten as a linear model. Although, even the margin score may show benefit since the use of bottom-up prediction in training can change what weights are learned. In Figure 10, we see that the nonlinear transform is in fact important and seemingly more so in the middle levels. In the future, we intend to investigate whether non-linear decision surfaces arise more often in the middle-level categories.

5. FUTURE WORK

There are many potential areas for future work, but we are most excited about casting the Refined Experts model that has been learned as a value-of-information problem and short-circuiting the bottom-up predictions based on model weights. Alternatively, we could use the search method of [24] as described in Section 2. An additional interesting area is to use other base learners. One natural choice would be learners that optimize a different performance measure (*e.g.* graph distance).

6. SUMMARY

In summary, we identified two key problems of hierarchical classification, error propagation and non-linear decision surfaces, that have gone largely unstudied. In addition to deepening the understanding of these phenomena, we introduced methods based on these insights, Refinement and Refined Experts, that significantly outperform a competitive baseline. In the case of Refinement, performance gains of 8-13% can be achieved with a training complexity on par with the baseline and a prediction complexity identical to it. When a more computationally intense method can be supported, Refined Experts provides even greater gains of 20-28% over the baseline by using "rough" guesses before refining them. Finally, we presented a series of ablation studies that help to not only verify important modeling decisions in our approach but also highlight important properties of hierarchical classification for future researchers to consider.

7. REFERENCES

[1] P. N. Bennett, S. T. Dumais, and E. Horvitz. The combination of text classifiers using reliability indicators. *Information Retrieval*, 8(1):67–100, 2004.

[2] C. M. Bishop and M. Svensén. Bayesian hierarchical mixtures of experts. In *UAI '03*, 2003.

[3] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM '04*, 2004.

[4] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Hierarchical classification: combining bayes with svm. In *ICML '06*, 2006.

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni. Incremental algorithms for hierarchical classification. *Journal of Machine Learning Research*, 7:31–54, 2006.

[6] O. Dekel, J. Keshet, and Y. Singer. Large margin hierarchical classification. In *ICML '04*, 2004.

[7] S. Dumais, E. Cutrell, and H. Chen. Optimizing search by showing results in context. In *CHI '01*, 2001.

[8] S. T. Dumais and H. Chen. Hierarchical classification of Web content. In *SIGIR '00*, 2000.

[9] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *ECML '98*, 1998.

[10] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[11] A. R. Klivans and A. A. Sherstov. Improved lower bounds for learning intersections of halfspaces. In *COLT '06*, 2006.

[12] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *ICML '97*, 1997.

[13] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[14] W. Li and A. McCallum. Pachinko allocation: Dag-structured mixture models of topic correlations. In *ICML '06*, 2006.

[15] T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explorations*, 7(1):36–43, 2005.

[16] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML '98*, 1998.

[17] D. M. Mimno, W. Li, and A. McCallum. Mixtures of hierarchical topics with pachinko allocation. In *ICML '07*, 2007.

[18] Netscape Communication Corporation. Open directory project. http://www.dmoz.org.

[19] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.

[20] M. E. Ruiz and P. Srinivasan. Hierarchical neural networks for text categorization. In *SIGIR '99*, 1999.

[21] S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-GrAdient solver for svm. In *ICML '07*, 2007.

[22] A. Sun and E. Lim. Hierarchical text classification and evaluation. In *ICDM '01*, 2001.

[23] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

[24] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *SIGIR '08*, 2008.

[25] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR '99*, 1999.

[26] B. Zhang, H. Li, Y. Liu, L. Ji, W. Xi, W. Fan, Z. Chen, and W.-Y. Ma. Improving web search results using affinity graph. In *SIGIR '05*, 2005.