Finding Your Way, Courtesy of Machine Learning

Chaitanya Asawa
Department of Computer Science
Stanford University
Palo Alto, CA 94305
casawa@stanford.edu

Marcus Gomez
Department of Computer Science
Stanford University
Palo Alto, CA 94305
mvgomez@stanford.edu

Viraj Mehta
Department of Mathematics
Stanford University
Palo Alto, CA 94305
virajm@stanford.edu

Abstract

In this paper, we considered the multi-label, multi-class prediction problem in the context of classifying undergraduate education requirements at Stanford University – namely, given a course's description, determining what general education requirements it satisfies. Given the extremely tough challenge of using a small dataset, we opt to consider models and make modifications that are able to handle multiple classes and produce multiple labels with very limited training examples. We also present a modification to the standard Naive Bayes algorithm that leverages Word2Vec style negative subsampling to more effectively leverage limited data.

1 Motivation

We believe that there is a flaw in the undergraduate graduation requirements: if a student makes the interesting choice of fulfilling his/her Formal Reasoning WAYS Requirement with Lie Algebra, as of now, he/she will not be able to graduate. Clearly, the WAYS requirements are currently not assigned completely and/or sufficiently.

2 Background

For undergraduates, Stanford requires completion of a series of requirements known as Ways of Thinking/Ways of Doing (WAYS). Undergraduates must take 11 courses across 8 different WAYS. These 11 courses are not fixed, and multiple courses can fulfill a given WAY. A course can also fulfill multiple WAYS.

3 Problem

The problem we would like to explore is as follows: given a course's description, if we know that the course satisfies at least one WAYS, can we predict the WAY(S) that it satisfies? This is a multi-label, multi-class problem as the multiple WAYS represent multiple classes, and the multiple WAYS a course could satisfy represent multiple labels.

As the WAYS system has only been recently introduced, not all courses have been approved to or marked as satisfying a WAY. We believe that this could be of interest as a tool for the Registrar and may take steps to make it available to them as part of a maturing WAYS program.

In our particular case, this problem is extremely challenging in that we have very little data (our motivation is to help more data of this form be possible), and yet, we need to be able to predict multiple classes and labels with limited training knowledge.

4 Related Work

To our knowledge, no group has worked on this problem with this dataset. Additionally, to date, multi-label+multi-class problems have not been extremely well-researched due to complexity. There are a few modern standards for said problem class. Schapire and Singer developed BoosTexter [1], an extension of AdaBoost (well-studied boosting algorithm). Elisseeff and Weston developed a generalized version of SVMs that minimizes a rank-based loss instead of pairwise binary losses [2]. Zhang and Zhou developed a multi-label generalization of KNN (dubbed ML-KNN) that identifies neighbors and chooses a label set using maximum a posteriori (MAP) estimation [3].

5 Data

Our data will come from Stanford's ExploreCourses, which for a given course, provides both a full course description and a list of WAY(S) that it satisfies. We were granted access to this dataset.

6 Preliminary Results

6.1 Data Processing

We have managed to collect the course descriptions and WAYS of 14336 courses from ExploreCourses.

Out of these 14336 courses, only 1571 unique courses had WAYS. (When we say unique courses, we only took one course from a set of courses with the same description, to avoid biasing the data with duplicates).

These courses collectively satisfy 2085 WAYS, meaning for courses that do satisfy WAYS, they satisfy on average 1.33 WAYS.

As the unlabeled courses could qualify for satisfying some WAY, but are currently not marked as such, we have not in-

cluded them in our data, and focused only on courses that do have WAYS.

Then, we removed stop words (frequent words) as they typically do not provide any insight into the text due to their high frequency and presence in every class. We also converted all words to lowercase and removed all punctuation. To assure that all words with the same stem but different suffix were treated similarly (as they have the same intention for the most part), we also stemmed our data. Finally, we tokenized the data, because our models work on the word-level at the moment.

We then performed a random 90/10 split of the formatted data to constructed a training set and dev/test set.

6.2 Error Metrics

There are two error metrics that we have been using to gauge our performance in a loss-function-independent manner between classifiers.

The first, the False Negative Metric, is a naive approach that is useful for independent classifier methods but is too simple for the multi-label concept. We train an independent classifier for each WAY. For each training example, there are 1 or more WAY labels. For each given test label, we run the relevant classifier. If the classifier misclassifies the example, we increment the error count. The metric is then given by

$$D_n = \frac{\text{number of misclassified labels}}{\text{number of labels}}$$

The second metric is Hamming Distance, calculated as follows. Take a training example $x^{(i)}$. Let $A^{(i)}$ be the set of true WAYS of the example and let $B^{(i)}$ be the set of labeled ways by our classifier. Then the Hamming distance $D_h^{(i)}$ is computed as

$$D_h^{(i)} = 1 - \frac{|A^{(i)} \cap A^{(i)}|}{|A^{(i)} \cup B^{(i)}|}.$$

The average of these $D_h^{(i)}$ over each training example. is our overall metric.

By using two metrics, we have a complete picture of the performance of our classifiers on a single-label and multi-label classification test. In addition, the two-metric system allows us to dig deeper into the methods that are inherently multi-label as opposed to those that use a combination of binary classifiers.

6.3 Baseline Models

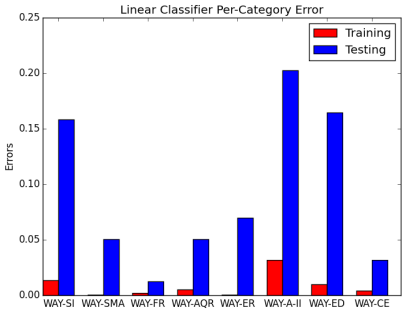
In our baseline, we decided to construct models which involve developing a binary classifier for every single WAY. This would mean we have 8 classifiers, each indicating whether or not a course satisfies a particular WAY. We would train the classifiers individually. Then, at test time, we would run each classifier over a course description, and determine which WAYS the course satisfies. We have constructed three such sets of models.

6.3.1 Linear Classifier

We constructed a standard linear model, using word frequency counts from descriptions as features.

We were able to achieve 0.0057 False Negative metric on a training set and 0.0973 error on a dev/test set. Additionally, using the aforementioned Hamming Distance we achieved 0.399 as our error.

To better understand our performance, we performed a fine-grained analysis over how each WAY classifier was doing. We found that some of the more STEM-oriented WAYS (such as Formal Reasoning (FR), Applied Quantitative Reasoning (AQR), and Scientific Method Analysis (SMA)) had better performance than other WAYS.



6.3.2 Naive Bayes

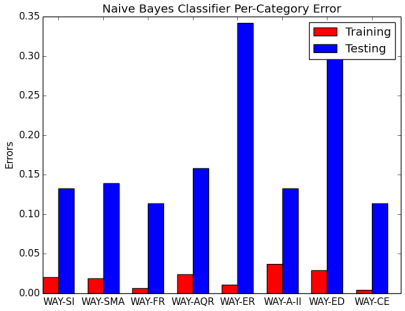
6.3.2.i Word2Vec Style Subsampling

We constructed a Naive Bayes model from scratch. To account for the small amount of training data and the skewed proportions of the positive and negative examples, we constructed an analog of the negative subsampling method used by Mikolov et. al in their loss function [4]. In particular, we randomly sampled the negative training examples so that there was an equal number of positive and negative examples in each class. Then we averaged the parameters determined by the Naive Bayes algorithm on all of these random iterations. Though this is not a standard part of the Naive Bayes procedure, and we could not find any literature examples of this being done to improve performance, this gave us a significant improvement in both error categories (on the order of 30%).

6.3.2.ii Results

We were able to achieve 0.02 False Negative error on a training set, and 0.12 False Negative error on a dev/test set. Using the Hamming Distance metric of error, we achieved an error of 0.507. This (in addition to the proceeding figure) seems to indicate we have a series of reasonably good single-WAY classifiers, but their combined performance is not as well.

To better understand our performance, we performed a fine-grained analysis over how each WAY classifier was doing. We see that WAY-ER and WAY-ED perform exceptionally poorly, whereas all the other WAYS perform reasonably the same.



6.3.3 Support Vector Machines

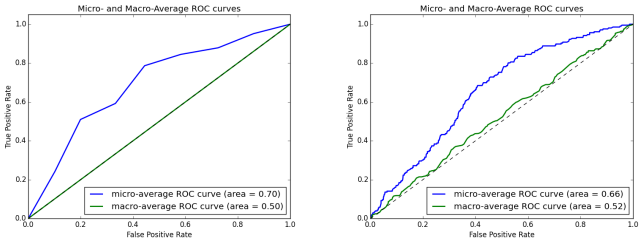
In addition to the baseline models, we leveraged the previous baseline approach using support vector machines (SVMs). In particular, we used a linear kernel of the form:

$$K(x_i, x_j) = x_i^T x_j$$

and a radial basis function kernel of the form

$$K(x_i, x_j) = \exp(-\frac{||x_i - x_j||_2^2}{2\sigma^2})$$

For the linear kernel, we achieved a training error of 0.000, and a test error of 0.823 (error here is average hinge loss); for the RBF kernel, we achieved a training error of 0.000 and a test error of 1.000. The models we implemented here seem to indicate that SVMs are a poor model for the multi-class multi-label problem. As further, evidence, we show the ROC curves below (left is linear, right is rbf). The macro-average ROC curve is generated by considering the average of the ROC over each class; the micro-average curve is generated by considering all positives and negatives as a collective single set. Note importantly that for both the macro- and micro-average ROC curves, the area under said curves is less than 0.7 and in the micro-average case, the curve is effectively linear; this is a strong indicator that in general, the SVMs are not expected to rank a randomly chosen positive and randomly chosen negative sample any differently (i.e. the SVM test is effectively random/worthless).



7 Forward Steps

The common thread through our simpler models is that there are reasonably good binary classifiers that we are not doing a great job combining well with each other for a multilabel model. We are looking into trying the multi-class boosting methods detailed by Zhu et al. as well as different problem transformation strategies.

Looking forward, we would like to experiment with using neural network models, which are typically considered more powerful than the models we have implemented so far. These models are able to capture potentially non-linearities that our current models have trouble expressing.

When using these neural network based models, we will potentially vary between different architectures and try to tune the models appropriately with features such as the activation function, number of hidden units, and so on.

Finally, we would like to get in contact with the Stanford University Registrar and see if this is a project that may be useful to them as a class-processing tool.

Acknowledgments

Thank you ExploreCourses for giving access to course data.

References

[1] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," in *Machine Learning*, pp. 135–168, 2000.

[2] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *In Advances in Neural Information Processing Systems 14*, pp. 681–687, MIT Press, 2001.

[3] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recogn.*, vol. 40, pp. 2038–2048, July 2007.

[4] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.