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 consider 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 present a slightly modified version of Naive Bayes, an application of BoosTexter, and several iterations of linear classification techniques. Then, we explore the performance of techniques from deep learning including RNN, LSTM, and GRU.

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 WAY, 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 Generalizability

Part of the motivation for studying the multi-class, multi-label classification problem in this setting (using only a course description and not necessarily all other tags) was that we would be able to explore the problem in a generalizable manner that we could then use for other problems. In order to create a more general multi-class, multi-label text classification module, we restrict our analysis to descriptions, using word frequency features only. Later, we describe some potential improvements from using dataset-specific features as mentioned in section 11. Since we were interested in studying the multi-class, multi-label problem in general, adding hand-engineered features would decrease the portability of the models and especially the neural models attempt to gather that information as a part of the process.

7 Preliminary Results

7.1 Data Processing

We have collected 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 included 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. The data was not stemmed for the deep learning models, in which we do not stem the words since we are using word vectors as inputs. 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.

7.2 Error Metrics

There are two error metrics that we have been using to gauge our performance in a loss-function-independent manner between classifiers. We use the first metric only for baseline methods, when we have multiple classifiers. The second metric, the Hamming distance, is the metric we will use to ultimately evaluate all 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(s). Then the Hamming distance $D_h^{(i)}$ is computed as

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

The average of these $D_h^{(i)}$ over all training examples 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.

7.3 Baseline Models

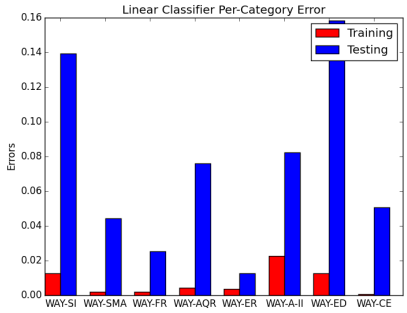
In our baseline, we decided to construct models which involve developing a binary classifier for every single WAY. This would mean we have 8 classifiers, each indicating whether or not a course satisfies a particular WAY. We would train the classifiers individually. Then, at test time, we would run each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

7.3.1 Linear Classifier

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

We were able to achieve 0.0039 False Negative metric on a training set and 0.0466 error on a dev/test set. Additionally, using the aforementioned Hamming Distance we achieved 0.311 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.



7.3.2 Naive Bayes

7.3.2.i Word2Vec Style Subsampling

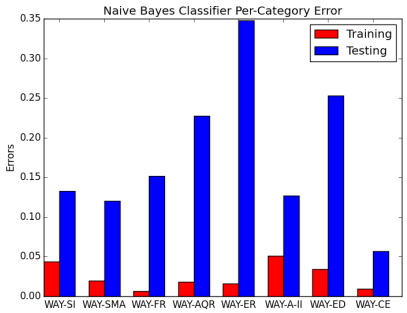
We constructed a Naive Bayes model from scratch. To account for the small amount of training data and the skewed proportions of the positive and negative examples, we constructed an analog of the negative subsampling method used by Mikolov et. al in their loss function for word2vec [4]. In particular, we randomly sampled the negative training examples so that there was an equal number of positive and negative examples in each class when training. We performed this subsampling for a few iterations, and then we averaged the parameters determined by the Naive Bayes algorithm. 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%).

7.3.2.ii Results

We were able to achieve 0.0068 False Negative error on a training set, and 0.0214 False Negative error on a dev/test set. Using the Hamming Distance metric of error, we achieved an error of 0.487. 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.



7.3.3 Support Vector Machines

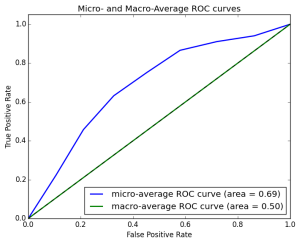
In addition to the baseline models, we leveraged the previous baseline approach using support vector machines (SVMs) [5]. In particular, we used a linear kernel of the form:

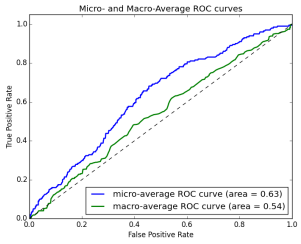
$$K(x_i, x_j) = x_i^T x_j$$

and a radial basis function kernel of the form

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

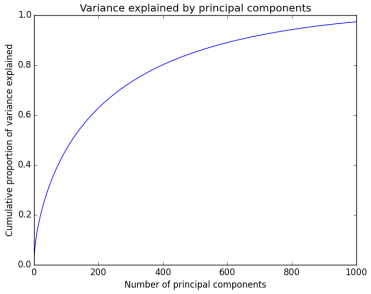
For the linear kernel, we achieved a training error of 0.000, and a test error of 0.823 (error here is average hinge loss); for the RBF kernel, we achieved a training error of 0.000 and a test error of 1.000. The models we implemented here seem to indicate that SVMs are a poor model for the multi-class 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).



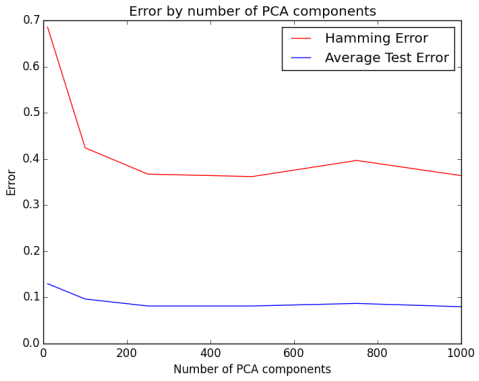


8 Dimensional Reduction via Principal Components Analysis

Work by Robles et. al. indicates that both for general text-classification tasks and for specifically multi-label text-classification tasks, dimensionality reduction techniques may help reduce model variance and improve overall performance, especially in the context of high vocabulary/feature size (in our case, V $\approx$ 9000) [6]. Thus, we used Principal Components Analysis (PCA) to reduce the number of features. As shown below, we explain most of the variance with around 1000 features, or $1/9$th of the dataset

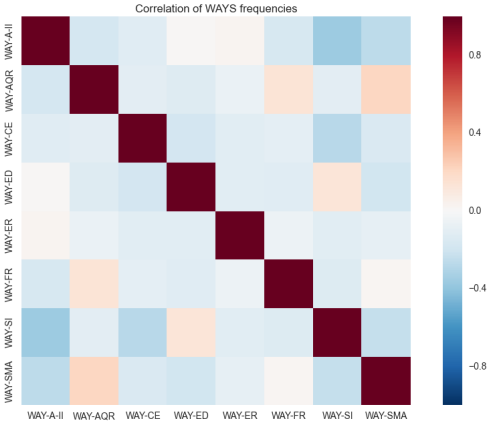


Running feature reduction, however, did not yield significant improvements in overall model efficacy; after the first 10 principal components, for both hamming error and average per-class testing error error plateaus and no additional components yield large decreases in error.



9 Class Independence Assumption

In the OneVsRest style models, we make the assumption that the classes are all independent. However, this assumption may not be true, and inter-class dependence may yield important interactions not accounted for by just the descriptions. In particular, consider the correlation matrix of WAYS frequencies shown below (obtained using Graphical Lasso since the dataset is fairly small)

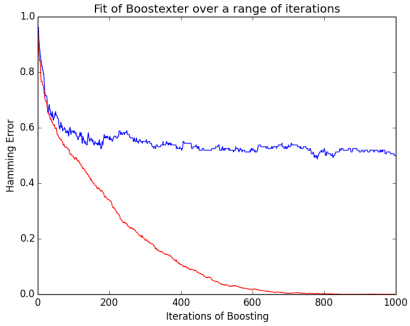


	A-II	AQR	CE	ED	ER	FR	SI	SMA
A-II	1.0	−0.178	−0.121	0.012	0.024	−0.168	−0.367	−0.264
AQR	-0.178	1.0	-0.106	-0.129	-0.065	0.140	-0.102	0.218
CE	-0.121	-0.107	1.0	-0.182	-0.110	-0.101	-0.281	-0.150
ED	0.012	-0.129	-0.182	1.0	-0.114	-0.122	0.129	-0.194
ER	0.024	-0.065	-0.110	-0.114	1.0	-0.061	-0.110	-0.090
FR	-0.169	0.140	-0.101	-0.122	-0.061	1.0	-0.137	0.019
SI	-0.367	-0.102	-0.281	0.129	-0.110	-0.137	1.0	-0.236
SMA	-0.264	0.218	-0.150	-0.194	-0.090	0.019	-0.235	1.0

Note that there are clearly non-trivial inter-class correlations (e.g. SI (Social Inquiry) and A-II (Artistic and Aesthetic Inquiry) are anti-correlated, SMA (Scientific Method and Analysis) and A-II are anti-correlated) that simply do not get accounted for in the OneVsRest models.

10 BoosTexter

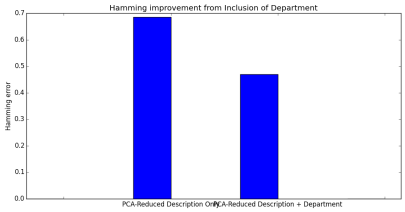
We noticed the clear interclass correlations and nonlinearity of features, suggesting we try something that can easily capture these nonlinearities. BoosTexter is an extension of Adaboost that is well-suited to multi-class/multi-label learning and text classification [7]. At a high level, BoosTexter swaps the decision stumps which return a binary value for a series of real-valued weak learners that allow for a ranked list of label likelihoods. In particular, we used a modified version of the icsiboost framework[8].



We found that boosting methods exhibit the standard behavior of overfitting on our dataset. It seems that we need more data so that we can have better test performance. At the point where the training and test errors diverge, we have a hamming loss of 0.56. We believe that the other reason boosting doesn't perform well is that there isn't a way to encode prior knowledge of the terms in the text. (As an example, 'Kant' is a term that would immediately lead one to an Ethical Reasoning label, but our software is unlikely to have learned such an association.)

11 Improvement with Domain-Specific Features

Although we restricted ourselves to word frequencies to create a general use tool, for this particular dataset, there were some features independent of raw text that could have some predictive power. As an example, below we show result of adding the course department as a feature in addition to the first 10 principal components in a linear classifier. Similar results were seen for other methods as well.

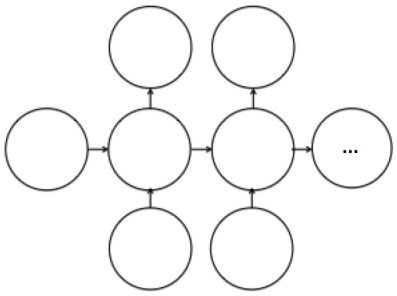


Another feature we could have added was the old DB, EC, or IHUM requirement labels, but we considered that this was not a useful feature for future data. It also would make our classifiers highly nonportable to other problems.

By using only descriptions for multi-label, multi-class classification, we focused on experimentation with models that could hopefully apply generally, rather than hand-engineering features.

12 Neural Sequential Models

As our other models were working on the token level, they were not able to necessarily capture the non-linear interactions between words. To handle this, we experimented with Neural Sequential Models. These models generally take a sequence of inputs and try to somehow capture properties about the sequential nature of the data.



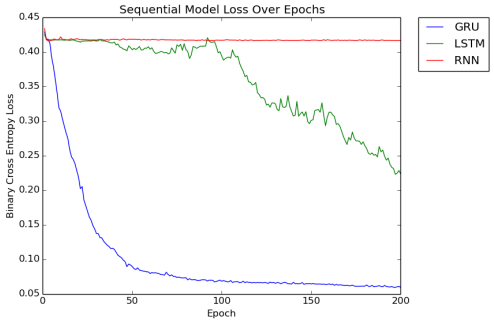
A general recurrent neural network, where the bottom layer of nodes are inputs, the middle layer are hidden states, and the top layer are output states.

We experimented with 3 sequential models with Keras [9]:
-RNN: Recurrent Neural Network
-LSTM: Long-Short Term Memory Network
-GRU: Gated Recurrent Network

In all of these models we applied a RELU layer followed by a softmax layer on the last output of the sequential model. The output vector of the softmax layer served as a probability distribution for each of the 8 possible WAYS, and we tuned a parameter that would determine some probability threshold for whether or not a course deserved a particular WAY label. We used binary cross entropy loss for our loss function.

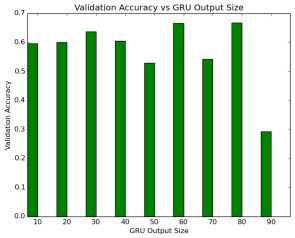
We used word vectors released by Stanford that were created using the GloVe method as inputs to our sequential models [10].

For the 3 models experimented with, we plot their loss over time:



It seems that a simple RNN did not change the loss over time. One hypothesis for this is a Vanishing Gradient problem, as described by Bengio et. al [11]. In particular, since the number of inputs to the RNN is the number of words in a course description, and in our data descriptions consist of 53 words on average, the RNN consist of 53 hidden states.

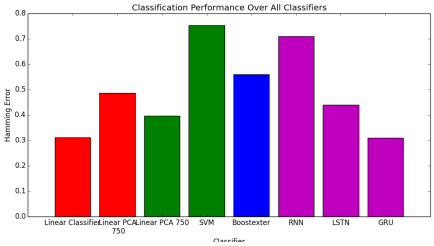
After seeing the superior performance of the GRU, we focused on tuning the GRU. One hyperparameter we tuned was the output size of the GRU, before applying the softmax layer.



We found that changing the output size, at small sizes, had little effect on accuracy. We hypothesize that despite there being many inputs, the GRU does not utilize too much information and hence does not have a much different accuracy with higher dimensional embeddings. A high dimensional output, however, seems to harm performance, potentially due to more unnecessary parameters.

13 Conclusion

We compare the performance of all of our models:



We see that of all our classfiers, the best two were the linear classifier and the Gated Recurrent Network. The GRU performed slightly better than our linear classifier. While in practice GRUs tend to typically have superior performance to linear classifiers, we believe the dearth of data prevented the GRU from learning its plethora of parameters well.

14 Forward Steps

We believe that our most limiting factor was the lack of data available to us. We believe that with more examples available (as they will be over time as the WAYS program matures), our various methods will be able to increase their effectiveness.

Beyond research, we also reached out to the registrar for their potential interest in such a tool, as the WAYS program matures.

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.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[6] P. J. M. Zhang, Min-Ling and V. Robles, "Feature selection for multi-label naive bayes classification," vol. 179, pp. 3218–3229, 2009.

[7] R. E. Schapire and Y. Singer, "Boostexter: A boosting-based system for text categorization," *Machine learning*, vol. 39, no. 2, pp. 135–168, 2000.

[8] B. Favre, D. Hakkani-Tür, and S. Cuendet, "Icsiboost." `http://code.google.come/p/icsiboost`, 2007.

[9] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[10] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[11] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.