Bryce Poeppel – >(John Doe)

Data Mining

Data Mining Competition Final Report

**Problem definition:**

PubMed is a public database that hosts a large variety of biomedical literature to be used for research. PubMed has developed its own system of tagging articles, MeSH or Medical Subject Headings, with their associated labels from the US National Library of Medicine. The principle goal of the MeSH vocabulary is to make the articles both easier to tag then then easier to be indexed and searched.

The problem arises from the size of the data base as well as the variety that exists within the database. PubMed claims to host over 29 million articles coming from a rather diverse set of fields such as biomedical, life science and other online literature. To label all of this data it was previously solely dependent on individuals tagging the articles manually. With the rising popularity of machine learning and related topics there became a desire to at least partially automate the labeling process. Not only for the sake of time but also for accuracy as human entry of the MeSH terms is occasionally prone to errors.

The goal of this class’ competition is to explore potential implementations that could automate the labeling process. This constitutes a multiclass multilabel classification problem as each article can be assigned a collection of labels.

**Literature Review:**

From the literature there seem to be two modes to approach multilabel classification. One such mode is to consider the labels as disjoint occurrences and to then perform classification of each sample once for each label. The other idea is to develop the labels into classes themselves and assign the sample a collection of labels rather than to consider each label discretely. Paper [1] suggests that mode 2 might be the best for situations in which each sample is to be assigned more than an arbitrarily small number of labels. This idea seems to be supported by a few of the other papers but small is not well defined.

Most of algorithms designed to deal with multilabel and multiclass problems work to force the problem into one or the other form. This is generally done by giving favor to either the labels or the classes such that there are fewer moving pieces within the algorithm. One such method is to consider that each sample is assigned at most one class; this would mean that there would need to be many classes to cover the many possible combinations. Or it could be considered a multiclass problem in that each sample can be assigned a number of classes and the classes each have at most one label. Or there could be many classes assigned to each sample each with many labels. All of these have some issues, generally related to how many possible combinations must exist to cover all cases.

The literature also suggests that many of these algorithms might require some form of dimensionality reduction, especially as the number of combinations possible in the algorithm rises. Though PCA is a possible form of dimensionality reduction [4] and [5] suggests term frequency inverse document frequency as a better means for reduction.

**Methods and Results:**

The first thing that was done for this project was to run many classifiers without doing any optimization to find which one is most likely to be best. Given that it was discussed in class that getting an F1 score of .6 was expected for most and that .75 would probably be high I set out trying for get close to .6 and then planning to optimize whichever got closest. So I started by trying Naïve Bayes Multinomial, Naïve Bayes Gaussian, K Nearest Neighbors, Linear SVM and Logistic Regression. I used sklearn.metrics.f1\_score to generate the expected F1 scores using k fold revalidation where k = 3.

|  |  |
| --- | --- |
| Method | Estimated F1 |
| NB Multinomial | 0.28 |
| NB Gaussian | 0.25 |
| KNN | 0.34 |
| SVM | 0.56 |
| LR | 0.5 |

From this table it was decided that SVM was likely the best of the methods that I first tested. Then I made the decision, without really thinking about it, to try TFIDF. This probably happened because the TA mentioned that this would likely be the best of the dimensionality reduction techniques for this competition. I also saw that in the homework 2 for the Machine Learning course included some dimensionality reduction on the data set. So I decided it would be best to try.

From this table I, eventually, realized that the dimensionality reduction was hurting my results more than helping them. So I decided that TFIDF was out. I had also heard it discussed in class that PCA was a horrible idea as it would mask the rare classes so I didn’t attempt that. SVD is the generalized form of PCA so I am not sure what possessed me to try it but when I did the expected F1 score was so bad I didn’t bother recording it.

Given that normalization and standardization is carried out as a default in the sklearn implementation. The next step was to find which option of the SVM was likely to be best if it was not –s 2. I tried –s 3 and it told me to try –s 2. I was suggested in class that –s 2 was likely the best. I looked at the options above –s 4 and didn’t understand them and couldn’t find the documentation for them for comparison so I settled on –s 2. Then the next step was to find the optimal c value.

|  |  |  |
| --- | --- | --- |
| c | Estimated F1 | Actual  F1 |
| 5 | 0.6 |  |
| 1 | 0.63 |  |
| 0.05 | 0.68 |  |
| 0.001 | 0.76 | 0.74005 |
| 0.005 | 0.773 | 0.74288 |
| 0.0005 | 0.64 |  |

From this table it was decided that the most optimal c value would likely be .005 or .001. This table has been adjusted to use the F1 score and not the F0 score, which I accidentally lost, but I do recall being quite surprised that the F0 scores were much higher than the F1 estimates on average. The F0 scores for these two was somewhere in the ball park of .76-.79. Which I was quite content with as it put me high on the leaderboards but I decided it was time to try something else. Once the F0 -> F1 fix on the leader board c = .005 ended up being my best competition score, though at the time I was unaware of this. I was beginning to notice that the estimated F1 scores were not always good estimates but I assumed it had to do with my resampling method. Before I was using K Fold with k = 3 but I switched to bootstrap after these tests assuming that K fold was merely picking up samples that were too close to one another within the dataset.

I moved on to decision trees for the next method as it was the final method I mentioned wanting to try in my proposal. As with the methods when I established my baseline I decided to try the Decision Tree without any parameter optimization first. I lost my first attempt unwittingly as I forgot to save my predictions into the matrix I believed was storing them resulting in a 0.0 F0 submission. So I tried again. This method took almost 16x longer to complete than SVM and much longer than the other more basic approaches, clocking in at somewhere around 12 hours. It did however move me up the leaderboard before the switch from F0 to F1 allowing me to break into the top 10 with F0 of ~.81. After the results tabulation issue was fixed though this method ended up having an F1 score of .65711, much lower than I was aware of. I then ran if once more with parameters; min samples to split = 500, max features = 140 and got an extremely small change in F1 but a better run time saving around 2 hours.

Now in the top 10 I decided I would try Random Forest since it was more likely to produce better results. Fearing that it would take even longer than the Decision Tree I wanted to start working on it sooner as it is predicted that Random Forest should be capable of putting up better results than the Decision Tree. I ran Random Forest with only one change to the parameters, min samples to split = 500 since this did not seem to impact my results for the Decision Tree and made it run slightly faster. This method took nearly 16 hours. It produced some an extremely good F0 score though of approximately .83. This encouraged the next run where the parameters were; min samples to split = 500, max features = 140, and number of trees was 50. The resulting F0 was .8343 which became my highest submission. This run unfortunately took almost 40 hours, not because it ran the whole time though. A windows update changed the time my computer slept from never to 45 minutes. I started it in the morning left for work then came home to find it had stopped. Assuming someone in my home turned it off after seeing my computer running I started it again and went to bed. I only realized the issue after waking up to it again stopped. Again I noticed that my estimated F1 was much lower than I was getting when I submitted the predictions for these executions but was at that point I was unsure why. Now that the scores are updated and I see that the F1 scores are .63452 and .65076 respectively and I regret having used so much time on these methods.

Once that was sorted the time of the competition was running down so I decided to try out ensemble classification. I decided on majority voting as it is the easiest. For the first submission I tried my best SVM, Decision Tree and Random Forest, the resulting F0 score was worse than my best submission at the time, which was the second Random Forest run. So I tried again using my two best SVMs, the Decision Tree and two best Random Forest predictions. This was done to bias the results of the SVM and Random Forest over the Decision Tree. This resulted in my best submission of .8424 F0. With the F0 to F1 change I made one more ensemble submission using two of the best SVM, one Decision Tree, one Random Forest and one Logistic Regression. These are selected as they encompass the new best submissions with some bias toward the best found method of SVM. This gave me a new best of .74354 F1.

**Conclusion:**

SVM using option 2 one vs all classification with a c of .005 for all classes is the best standalone result that was obtained using the implementation provided within the Liblinear for Matlab package with an F1 of .74288. The best ensemble method was the combination of my two of the best SVM, one Decision Tree, one Random Forest and one Logistic Regression with an F1 of .74354. I operated under the false conclusion that Random Forest was the best classification method for quite some time due to the F0 not F1 issue. If I were to try this again I would like to attempt more ensemble methods, using different scoring metrics and also trying different combinations to see if any would yield better results. Also I would not have used so much time on Decision Trees and Random forest as I would have known that the F1 score from the original runs was much lower than what was presented to me.

**References:**

[1] R. Rak, L. Kurgan, and M. Reformat, “Multilabel associative classification categorization of MEDLINE aticles into MeSH keywords,” IEEE Engineering in Medicine and Biology Magazine, vol. 26, no. 2, pp. 47–55, 2007.

[2] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 9:1871–1874.

[3] Katakis, Ioannis, Grigorios Tsoumakas, and Ioannis Vlahavas. "Multilabel text classification for automated tag suggestion." In Proceedings of the ECML/PKDD, vol. 18. 2008.

[4] Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. "Bag of tricks for efficient text classification." arXiv preprint arXiv:1607.01759, 2016.

[5] Kibriya A.M., Frank E., Pfahringer B., Holmes G. (2004) Multinomial Naive Bayes for Text Categorization Revisited. In: Webb G.I., Yu X. (eds) AI 2004: Advances in Artificial Intelligence. AI 2004. Lecture Notes in Computer Science, vol 3339. Springer, Berlin, Heidelberg term frequency inverse document frequency