

# Machine Learning Methods for Biological Data Curation

Kelley Paskov (kpaskov@stanford.edu)

## Background

Biological publications are being produced more rapidly than any human could possibly read them, so modern biologists rely heavily on the manually curated data in biological databases. This data is collected by expert biocurators who read literature focused on a particular organism, disease, or process and extract key information in a structured format using ontologies and controlled vocabularies. We would like to automate parts of this process and focus on two prediction tasks:

- 1) Triage Task: Based on its abstract, can we determine if a new paper contains relevant information? This is a binary classification task where a label of 1 indicates that a paper contains relevant information and a label of -1 indicates that a paper does not contain relevant information.
- 2) Multi-label Classification Task: Assuming a paper has relevant content, based on its fulltext, can we determine which biological category (or categories) can be associated with it? Since each paper can be associated with multiple biological categories, this is not a multi-class classification task, but rather a multi-label classification task. Given  $k$  categories, for each paper we would like to predict a binary vector of length  $k$  indicating which biological categories the paper is associated with.

Some promising work has been done on automated paper triage and gene detection within text, particularly related to the BioCreative and BioNLP competitions aimed at evaluating the state of the art in biomedical text mining. But tools produced from these competitions often have too narrow a focus or are unable to handle datasets on the scale of well-established model organism databases.

## Data

We have collected the full text of 70,000 articles curated by the *Saccharomyces* Genome Database (SGD) over the past 20 years, and the multi-category label for each article along 5 biological categories (ie we have a multi-label classification task with  $k=5$  categories). We have also collected the abstracts from 150,000 articles, including the 70,000 above and an additional 80,000 articles that do *not* contain annotatable information.

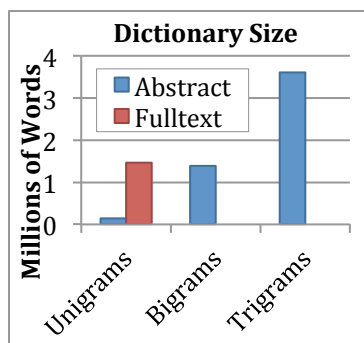
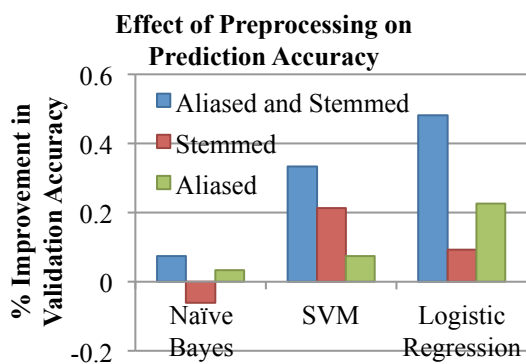


Figure 1: Feature set size.

## Features

We chose to represent each paper as a bag of  $n$ -grams, capturing the number of occurrences of each  $n$ -gram in the text of the paper. This representation is effective and is easy to compute over the large amounts of text in our two datasets - currently over 4 gigabytes. For our abstract corpus, we experimented with unigram, bigram and trigram<sup>1</sup> representations. For our fulltext corpus, we used only the unigram representation due to

<sup>1</sup> The bigram feature set contains both unigrams and bigrams, while the trigram feature set contains



**Figure 2: Effect of preprocessing on validation set accuracy relative to no preprocessing.**

the accuracy of our learning algorithms, we compared performance on three baseline unigram models to performance with no preprocessing (see Figure 2) and found that both aliasing and stemming words has a positive effect on performance.

## Models

We started by training four types of classifiers<sup>6</sup> on the triage task: logistic regression, SVM, Naïve Bayes, and KNN. Logistic regression and SVM outperform the other two models, and in fact perform nearly identically. We chose to continue our analysis of the multi-label classification task with logistic regression because logistic regression models the probability that a datapoint has a label of 1. This can be used as a confidence score, allowing us determine how confident we are in a given prediction. Because an SVM only takes into account the support vectors, a small subset of the training data, it is difficult to generate a meaningful confidence score for points that aren't support vectors. Furthermore, the logistic regression model allows us to easily tune the false positive/false negative tradeoff, an important factor for this task.

For our triage task, it's extremely important for our classifier to generate very few false negatives. False negatives correspond to missed information, which we would like to avoid. In order to drive the number of false negatives down, we made a modification to the logistic regression model. According to the logistic regression model, the probability that  $x^{(i)}$  has label  $y^{(i)} = 1$  is modelled as

$$P(y^{(i)} = 1 | x^{(i)}; \theta, b) = h_i(\theta, b) = \frac{1}{1 + \exp(-\theta^T x^{(i)} - b)}$$

All features are normalized by their  $L_2$  norm, i.e. we scale each feature so that  $\sum_{i=1}^n [x_j^{(i)}]^2 = 1$  for  $j = 1, \dots, d$ . We *did not* normalize individual documents because

computational constraints. Figure 1 shows the distribution of dictionary sizes for each of our feature sets.

We experimented with two data preprocessing steps: stemming and alias replacement. We used the porter2<sup>2</sup> stemming algorithm to stem all of the words in the corpus. We also wrote code to replace any biological terms or aliases with standardized names, including gene names and Gene

Ontology<sup>3</sup>, Yeast Phenotype Ontology<sup>4</sup>, and CHEBI<sup>5</sup> chemical terms. To determine whether these preprocessing steps increase

<sup>2</sup> <https://pypi.python.org/pypi/stemming/1.0>

<sup>3</sup> <http://geneontology.org/>

<sup>4</sup> <http://www.yeastgenome.org/ontology/phenotype/ypo/overview>

<sup>5</sup> <http://www.ebi.ac.uk/chebi>

<sup>6</sup> Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

they are all abstracts and have approximately the same length. We used  $L_2$  regularization, so the classical  $L_2$ -regularized logistic regression negative log-likelihood is

$$\ell(\theta, b) = - \sum_{i=1}^n [y^{(i)} \log h_i(\theta, b) + (1 - y^{(i)}) \log(1 - h_i(\theta, b))] + \frac{\lambda}{2} \|\theta\|_2^2$$

However, we are interested in a slightly different problem where we minimize the error on negative examples  $\mathcal{N} = \{1 \leq i \leq n | y^{(i)} = 0\}$  subject to the error on positive examples  $\mathcal{P} = \{1 \leq i \leq n | y^{(i)} = 1\}$  being below an acceptable threshold. This is given by the problem

$$\begin{aligned} \min_{\theta, b} \quad & - \sum_{i \in \mathcal{N}} \log(1 - h_i(\theta, b)) + \frac{\lambda}{2} \|\theta\|_2^2 \\ \text{subject to} \quad & - \sum_{i \in \mathcal{P}} \log h_i(\theta, b) \leq \tau \end{aligned}$$

It can be shown (through the Lagrangian) that, for an appropriately chosen  $\alpha \geq 0$ , the above is equivalent to minimizing

$$-\alpha \sum_{i \in \mathcal{P}} \log h_i(\theta, b) - \sum_{i \in \mathcal{N}} \log(1 - h_i(\theta, b)) + \frac{\lambda}{2} \|\theta\|_2^2$$

Therefore, we experimented with weighting our positive and negative classes in order to tune the number of false negatives on our validation set.

### Triage Task

For the triage task, we separated our corpus of abstracts into training, validation, and testing sets. We used the validation set to select a model and tune parameters and the testing set to report on the predicted performance of our optimal classifier.

We experimented with four different classifiers: naïve bayes, KNN, logistic regression, and SVM, our results are given in Table 1. Both logistic regression and SVM produced extremely similar models. This likely occurred because they both produce a linear decision boundary that takes into account each training example according to how well it is classified and can handle many correlated features because of  $L_2$  regularization. Both KNN and Naïve Bayes seemed to suffer from spurious and correlated features. This trend is most clear for KNN; as more features were added with the bigram and trigram datasets, performance steeply dropped off.

Table 1: Triage Task Results

(120,000 training samples; 15,000 validation samples; 15,000 testing samples):

| Model  | Corpus   | Features | Baseline | Training Accuracy | Validation Accuracy | Validation Sensitivity | Validation Specificity |
|--------|----------|----------|----------|-------------------|---------------------|------------------------|------------------------|
| NB     | Abstract | Unigrams | 53%      | 84.28%            | 81.69%              | .730                   | .891                   |
| NB     | Abstract | Bigrams  | 53%      | 86.90%            | 82.08%              | .734                   | .895                   |
| NB     | Abstract | Trigrams | 53%      | 89.65%            | 82.17%              | .734                   | .897                   |
| KNN    | Abstract | Unigrams | 53%      | -                 | 72.20%              | -                      | -                      |
| KNN    | Abstract | Bigrams  | 53%      | -                 | 56.00%              | -                      | -                      |
| KNN    | Abstract | Trigrams | 53%      | -                 | 59.30%              | -                      | -                      |
| SVM    | Abstract | Unigrams | 53%      | 89.57%            | 86.67%              | .832                   | .896                   |
| SVM    | Abstract | Bigrams  | 53%      | 94.19%            | 87.08%              | .841                   | .896                   |
| SVM    | Abstract | Trigrams | 53%      | 96.41%            | <b>87.22%</b>       | .842                   | .898                   |
| LogReg | Abstract | Unigrams | 53%      | 89.35%            | 86.51%              | .833                   | .893                   |

|        |          |          |     |        |               |      |      |
|--------|----------|----------|-----|--------|---------------|------|------|
| LogReg | Abstract | Bigrams  | 53% | 93.56% | 87.01%        | .841 | .895 |
| LogReg | Abstract | Trigrams | 53% | 95.66% | <b>87.17%</b> | .842 | .897 |

\* Baseline shows the accuracy of classifying each sample to belong to the most common training class.

Since curators must read all positively triaged papers, any false positives can easily be corrected during the annotation process. This means we are primarily interested in our classifier’s negative predictions. False negatives correspond to papers whose information is missed, which we want to avoid. Furthermore, since curators no longer need to read papers that are triaged as negative, the rate of negative predictions measures the time savings of our classifier. By changing the class weight parameter  $\alpha$  given in the equations above, we were able to tune the number of false negative samples on our validation set. Table 2 shows validation statistics using trigram features on the abstract corpus for the triage task. Although validation accuracy may decrease, tuning alpha allows us to decrease the expected number of false negative predictions, which results in a better classifier for the task.

Table 2: Alpha Tuning Results

| Alpha | Validation Accuracy | TP   | FN   | FP   | TN   | False Negative/<br>Predicted Negative | Predicted Negative/<br>All |
|-------|---------------------|------|------|------|------|---------------------------------------|----------------------------|
| 1     | 87.17%              | 5839 | 1092 | 833  | 7236 | 7.28%                                 | 55.52%                     |
| 0.1   | 85.99%              | 6054 | 877  | 1224 | 6845 | 5.84%                                 | 51.48%                     |
| 0.001 | 76.51%              | 6691 | 240  | 3283 | 4786 | 1.6%                                  | 33.51%                     |

Based on these results, we chose a logistic regression classifier using trigrams and an alpha parameter of 0.1 as our optimal classifier. Table 3 shows the predicted performance of this classifier based on the testing set.

Table 3: Testing Results on Triage Task

| Testing Accuracy | TP   | FN  | FP   | TN   | False Negative/<br>Predicted Negative | Predicted Negative/<br>All |
|------------------|------|-----|------|------|---------------------------------------|----------------------------|
| 85.8%            | 6330 | 687 | 1443 | 6540 | 4.58%                                 | 43.6%                      |

## Multi-label Classification Task

We split the fulltext corpus in a similar manner as the abstract corpus, creating training, validation, and testing sets. We chose to approach the multi-label classification problem by training 5 classifiers, one for each biological category. Results are shown in Table 4. Overall, this task seemed to be much more difficult than the triage task. This may have been due to the fact that our dataset had relatively few positive examples for each label. Performance for one of the five categories was fairly poor, with our logistic regression classifier performing less accurately than a simple baseline classifier that assigns every sample to the most common training class. Performance on the remaining three categories was adequate, but low. This task requires further work.

Table 4: Multi-label Classification Task Results

(70,000 training samples; 7,000 validation samples; 7,000 testing samples)

| Model  | Category | Corpus   | Features | Baseline | Training Accuracy | Validation Accuracy | Validation Sensitivity | Validation Specificity |
|--------|----------|----------|----------|----------|-------------------|---------------------|------------------------|------------------------|
| LogReg | GO       | Abstract | Unigrams | 88%      | 96.41%            | 82.67%              | .867                   | .575                   |
| LogReg | GO       | Abstract | Trigrams | 88%      | 100.00%           | <b>86.54%</b>       | .970                   | .395                   |
| LogReg | GO       | Fulltext | Unigrams | 88%      | 99.98%            | 85.23%              | .906                   | .507                   |
| LogReg | Phen.    | Abstract | Unigrams | 79%      | 97.23%            | 87.82%              | .913                   | .440                   |
| LogReg | Phen.    | Abstract | Trigrams | 79%      | 100.00%           | <b>91.58%</b>       | .987                   | .196                   |
| LogReg | Phen     | Fulltext | Unigrams | 79%      | 100.00%           | 90.55%              | .952                   | .300                   |
| LogReg | Phys     | Abstract | Unigrams | 91%      | 97.80%            | 91.54%              | .928                   | .654                   |

|        |      |                   |     |         |               |      |      |
|--------|------|-------------------|-----|---------|---------------|------|------|
| LogReg | Phys | Abstract Trigrams | 91% | 100.00% | <b>92.27%</b> | .983 | .567 |
| LogReg | Phys | Fulltext Unigrams | 91% | 99.78%  | 92.68%        | .945 | .707 |
| LogReg | Gen. | Abstract Unigrams | 91% | 100.0%  | 89.11%        | .921 | .625 |
| LogReg | Gen. | Abstract Trigrams | 91% | 100.00% | <b>92.38%</b> | .983 | .531 |
| LogReg | Gen. | Fulltext Unigrams | 91% | 99.79%  | 91.67%        | .946 | .644 |
| LogReg | Lit. | Abstract Unigram  | 77% | 97.75%  | 90.33%        | .730 | .941 |
| LogReg | Lit. | Abstract Trigrams | 77% | 100.00% | <b>90.85%</b> | .871 | .854 |
| LogReg | Lit. | Fulltext Unigrams | 77% | 99.45%  | 89.66%        | .760 | .923 |

\* Baseline shows the accuracy of classifying each sample to belong to the most common training class.

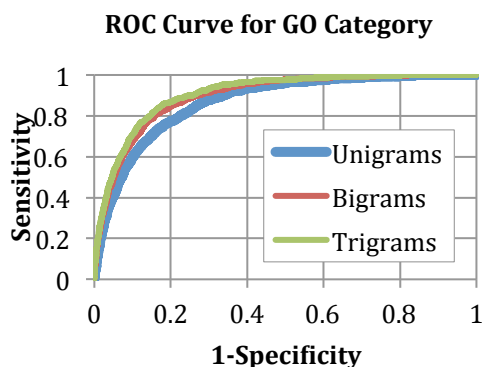


Figure 3: Sensitivity as a function of specificity for unigram, bigram, and trigram features for the GO multi-label category.

In Table 4, when comparing the performance of the abstract unigrams to the abstract trigrams, it appears that sensitivity increases while specificity decreases. However, this is because we are sampling different points along the ROC curve. When we plot the ROC curves for unigrams, bigrams, and trigrams (see Figure 3), we see that as we move to more complicated feature sets, the ROC curve improves. This is most likely because bigrams and trigrams have the power to capture informative phrases and multi-word biological terms that can't be captured by unigrams alone.

Based on these results, we chose to use the abstract trigrams dataset with a logistic regression model as our optimal classifier for this task. Table 5 shows predicted performance for the classifier for each category based on the testing set.

Table 5: Testing Results on Multi-label Classification Task

| Category | Baseline | Testing Accuracy | Testing Sensitivity | Testing Specificity |
|----------|----------|------------------|---------------------|---------------------|
| GO       | 88%      | 85.52%           | .934                | .386                |
| Phen.    | 79%      | 91.58%           | .974                | .192                |
| Phys.    | 91%      | 92.33%           | .966                | .640                |
| Gen.     | 91%      | 91.95%           | .965                | .511                |
| Lit.     | 77%      | 90.45%           | .701                | .946                |

## Future Work

There are several areas of this project that would benefit from future work. Due to the size of our datasets, we were only able to do parameter tuning at an order of magnitude scale. By tuning parameters at a more fine-grained level, we believe we could increase performance on both tasks. Our results on the multi-label classification task show that working with trigrams from the abstract corpus offers significant gains over working with unigrams. We believe that working with bigrams or trigrams from the fulltext corpus has the potential to dramatically increase performance by capturing information found in phrases and multi-word biological terms. However, this would require custom software because it would increase our feature size by at least an order of magnitude. Finally, the ultimate goal is to integrate our classifier into an active curation pipeline. We plan to present our results to the Saccharomyces Genome Database, and hope to eventually use these techniques to automate part of the biological data curation process.