DIMACS AT THE TREC 2005 GENOMICS TRACK

Aynur Dayanik, Alex Genkin, Paul Kantor, David D. Lewis†, David Madigan
{aynur,agenkin,paul.kantor,dmadigan}@rutgers.edu
davelewis@daviddlewis.com
DIMACS, Rutgers University
David D. Lewis Consulting†

ABSTRACT

This report describes DIMACS work on the text categorization task of the TREC 2005 Genomics track. Our approach to this task was similar to the triage subtask studied in the TREC 2004 Genomics track. We applied Bayesian logistic regression and achieved good effectiveness on all categories.

1. TEXT CATEGORIZATION TASK

The Mouse Genome Informatics (MGI) project of the Jackson Laboratory[1] provides data on the genetics, genomics, and biology of the laboratory mouse. In particular, the Mouse Genome Database (MGD) contains information for the mouse system annotated from literature.

To find information on mouse genomics biology, MGI first automatically scans new scientific literature for records containing one or more of the words "mouse", "mice", and "murine". In a triage step, MGI personnel then check each article to see if it contains information appropriate for inclusion in MGD. The goal of this triage process is to limit the number of articles sent to human curators for more detailed analysis. The TREC 2005 Genomics track [4] defined a categorization task based on simplified versions of the MGI triage process. It consists of the triage subtask from the TREC 2004 Genomics track [3], which aims to identify articles for Gene Ontology annotation, as well as three other major topics of interest to MGI. This year's categorization task includes the following four categories:

- Alleles of mutant types,
- Embryologic gene expresession,
- Gene Ontology (from TREC 2004),
- Tumor biology.

For the TREC 2004, full text articles published in 2002 and 2003 by three major journals were obtained. Those articles containing "mouse", "mice", or "murine" were identified and separated into a training set (5,837 documents from 2002) and a test set (6,043 documents from 2003). The same data were used for the TREC 2005 categorization task.

The goal for task participants was to identify which of the articles from the test set had, during MGI's operational manual triage process, been chosen for furhter consideration by MGI curators for the four categories listed above. We can view this, for each category, as a binary text classification problem, with articles chosen for curation during the triage process being positive examples, and those rejected during triage being negative examples. Logs from MGI were used to produce relevance judgments for the task data. Table 1 shows the number of positive and negative examples in the train/test sets. Note that, for each category, the proportion of positive examples in the test set is similar to that of the training set.

[1]http://www.informatics.jax.org

Task	Data	#Positives	#Negatives	%Positives
A (alelle)	train	338	5499	6.15
A (alelle)	test	332	5711	5.81
E (exp.)	train	81	5756	1.41
E (exp.)	test	105	5938	1.77
G (GO)	train	462	5375	8.59
G (GO)	test	518	5525	9.37
T (tumor)	train	36	5801	0.62
T (tumor)	test	20	6023	0.33

Table 1: Proportion of positive/negative examples in the train/test sets.

The official effectiveness measure for the categorization task was the normalized utility measure. Since the number of positive and negative examples varies for the four categorization problems (Table 1), different utility coefficients were used for the utility measure for each category.

The normalized linear utility was computed as

$$\text{Normalized Utility} = \frac{\text{Utility}_{\text{Raw}}}{\text{Utility}_{\text{Max}}}$$

where

$$\text{Utility}_{\text{Raw}} = \text{Ur} \cdot \text{TP} - \text{FP},$$
$$\text{Utility}_{\text{Max}} = \text{Ur} \cdot (\text{TP} + \text{FN}).$$

Here TP, FP, and FN are defined in the contingency table in Table 2, and different utility coefficients, Ur, were used for each category:

$$\text{Ur} = \begin{cases} 17, & \text{for A (allele)}, \\ 64, & \text{for E (expression)}, \\ 11, & \text{for G (GO)}, \\ 231, & \text{for T (tumor)}. \end{cases} \quad (1)$$

These utility coefficients (Ur values) were determined based on MGI's current operation of triaging everything, see [4] for

	Relevant	Not relevant
Retrieved	True positive (TP)	False positive (FP)
Not retrieved	False negative (FN)	True negative (TN)

Table 2: Contingency table.

	Categories			
Situation	**A**	**E**	**G**	**T**
Perfect prediction	1.0	1.0	1.0	1.0
Predict with MeSH "Mice"	0.60	0.60	0.55	0.46
Best submitted run	0.87	0.87	0.58	0.94
Triage everything	-0.01	0.11	0.03	-0.30
Triage nothing	0	0	0	0
Imperfect prediction	-1.01	-0.88	-0.96	-1.30

Table 3: Boundary cases for the normalized utilities on the test set.

details. Note that the number of positive examples for the GO category is different from the 2004 data because MGI has updated its database since then. Therefore, its utility coefficient is also different from the one used last year, which was 20.

Table 3 shows the values of normalized utilities for the boundary cases on the test data set.

Although the official effectiveness measure was the normalized utility measure, we also considered the F1 measure (F-measure with equal weight on recall and precision) [9, 5] where

$$\begin{aligned} \text{Precision (p)} &= \text{TP}/(\text{TP}+\text{FP}), \\ \text{Recall (r)} &= \text{TP}/(\text{TP}+\text{FN}), \\ \text{F1} &= \frac{2*\text{r}*\text{p}}{\text{r}+\text{p}} = \frac{2*\text{TP}}{2*\text{TP}+\text{FP}+\text{FN}}. \end{aligned}$$

2. BAYESIAN LOGISTIC REGRESSION

Logistic regression models estimate the probability that an example belongs to a class using this formula:

$$p(y_i = +1|\boldsymbol{\beta}, \boldsymbol{x_i}) = \frac{\exp(\boldsymbol{\beta}^T \boldsymbol{x_i})}{1+\exp(\boldsymbol{\beta}^T \boldsymbol{x_i})} = \frac{\exp(\sum_j \beta_j x_{i,j})}{1+\exp(\sum_j \beta_j x_{i,j})}$$

where y_i encodes the class of example i (positive/relevant $= +1$, negative/nonrelevant $= -1$) and $x_{i,j}$ is the value of feature j for example i. The model parameters $\boldsymbol{\beta}$ are chosen by supervised learning, i.e. by optimizing some function defined on a set of examples for which manually judged values of y_i are known.

In our work, we adopt a Bayesian framework and choose the $\boldsymbol{\beta}$ that maximizes the posterior loglikelihood of the data,

$$l(\boldsymbol{\beta}) = (-\sum_{i=1}^{n} \ln(1+\exp(-\boldsymbol{\beta}^T \boldsymbol{x_i} y_i)) + \ln p(\boldsymbol{\beta}),$$

where $p(\boldsymbol{\beta})$ is, for each $\boldsymbol{\beta}$, the prior probability that $\boldsymbol{\beta}$ is the correct parameter vector. The prior $p(\boldsymbol{\beta})$ encodes what we believe are likely values of $\boldsymbol{\beta}$ before seeing the training data.

We trained and applied all logistic regression models using Version 2.04 of the BBR (Bayesian Binary Regression) package [2][2]. BBR supports two forms of priors: a separate Gaussian prior for each β_j or a separate Laplace prior for each β_j. (The overall prior is the product of the individual priors for feature parameters.) The key difference between the two is that Gaussian priors produce dense parameter vectors with many small but nonzero coefficients, while Laplace priors produce sparse feature vectors with most coefficients identically equal to 0.

2.1 Choice of Hyperparameter

The Gaussian and Laplace priors have two hyperparameters for each model parameter β_j: a modal value μ_j (the most likely prior value of β_j), and a regularization hyperparameter (σ_j^2 for Gaussian and λ_j for Laplace) that indicates how close to μ_j we expect β_j to be. For simplicity, our TREC work assumes all μ_j's are 0, and that the regularization hyperparameter is the same for all features. This leaves a single regularization hyperparameter to be chosen for the whole model.

We consider a fixed set of hyperparameter values, and choose the one that maximizes the cross-validated posterior predictive log-likelihood for each training set. The prior variances considered were

$$0.5, 1, 4, 9, 16, 25, 36, 49, 64, 100, 10000, 1000000, 100000000$$

for both Laplace and Gaussian.

2.2 Threshold Selection

Logistic regression models estimate the probability that the example is a positive/relevant example. We then must convert this probability to a binary class label by choosing a threshold. We tested two approaches to choosing a threshold for a categorization problem:

- MEE (Maximum Expected Effectiveness): Choose the threshold that maximizes the expected value of the effectiveness measure on the test set, under the assumption that the estimated class membership probabilities are correct and that the corresponding binary random variables are independent [5].
- TROT (Training set Optimization of Threshold): Choose the threshold that maximizes the effectiveness measure on the training set.

Both TROT and MEE were tested by cross-validation on the training data. MEE was found consistently better and so was used for all our runs. The MEE thresholds for the normalized utility effectiveness measure are different for each category:

$$p(y_i = +1) >= \begin{cases} 1/18 = 0.0555, & \text{for A (allele)}, \\ 1/65 = 0.0153, & \text{for E (expression)}, \\ 1/12 = 0.0833, & \text{for G (GO)}, \\ 1/232 = 0.0043, & \text{for T (tumor)}, \end{cases} \quad (2)$$

on a probability scale. We should note that these are very low thresholds by the standards of most text classification research.

[2]http://www.stat.rutgers.edu/∼madigan/BBR/

3. TEXT REPRESENTATION

The track provided the full text of the journal articles in both SGML and XML form. We used the XML versions from *train.xml.zip* and *test.xml.zip*. We also made use of additional descriptions of each article. The track files *train.crosswalk.txt* and *test.crosswalk.txt* specified the PubMed ID for each article. We used these IDs to obtain the MEDLINE record for each article either from the ad hoc track data from TREC 2004, or by downloading from PubMed.[3]

We used two representations for the training and test articles:

- **Full Text**: The union of text from the title (<*atl*>), subject (<*docsubj*>), abstract (<*abs*>), and body (<*bdy*>) XML elements of the article.
- **MEDLINE**: The MeSH terms, Medical Subject Headings, from the MEDLINE record (lines starting with *"MH - "* in ASCII text format), plus the union of text from the title (<*ArticleTitle*>) and abstract (<*Abstract*>) elements of that record. MeSH terms were converted to single tokens (Section 3.1) and so were kept distinct from the two text fields.

3.1 Text processing

For the full text articles, we extracted the contents of the specified XML elements for the particular representation (see above), concatenated those contents, and deleted all the internal XML tags. For <ArticleTitle>, we tokenized text at white space, deleted punctuation at the start and end of tokens, replaced token-internal punctuation with "xxx", and used the prefix "titlexxx" to distinguish the title features. For instance, title word "3-phosphatases" became titlexxx3xxxphosphatases. The rest of the text processing was done using the Lemur[4] utility ParseToFile, in combination with the Porter stemmer [6] supplied by Lemur and the SMART [7] stoplist.[5] This parser performed case-folding, replaced punctuation with whitespace, and tokenized text at whitespace boundaries. The Lemur utility BuildBasicIndex was used to construct Lemur index files, which we then converted to document vectors in BBR's format.

MEDLINE records were handled the same way, except that MeSH terms were converted to single tokens (e.g. replacing "Mice, Knockout" with "MHxxxMicexxxKnockout") before Lemur processing to force them to have a separate term ID than words.

3.2 Term Weighting

BBR requires text to be represented as vectors of numeric feature values. We used TFxIDF (term frequency times inverse document frequency) weighting [8], with IDF weights computed on the training instances only. We computed the weight of term t_j in document d_i, w_{ij}, by

$$w_{ij} = \begin{cases} (1 + log_e(f_{ij}))log_e \frac{N+1}{n_j+1}, & \text{if } t_j \text{ is present in } d_i, \\ 0, & otherwise. \end{cases}$$

Here N is the number of documents from which IDF weights are computed (the categorization training set, so $N = 5837$ for our official runs), f_{ij} is the frequency of term t_j in document d_i, and n_j is the number of documents containing term t_j. The way we computed IDF weights, called *Lookahead IDF*, is a version of IDF weighting that defines a value even for terms that do not occur in the training corpus. It can be viewed as including a future document being weighted in the set of documents used to define term weights for it, thus the name "lookahead". We applied cosine normalization to the resulting document vectors.

4. EXPERIMENTS

For each of our text categorization runs we trained four thresholded logistic regression classifiers, one for each of the four categories. Our runs used the following techniques:

- {a,e,g,t}DIMACSg9md: Representation: MEDLINE. Weighting: TFxIDF with cosine normalization. Prior: Gaussian. Hyperparameter Variance: 49, 16, 9, 49 for the categories A, E, G, and T, respectively.
- {a,e,g,t}DIMACSl9md: Representation: MEDLINE. Weighting: TFxIDF with cosine normalization. Prior: Laplace. Hyperparameter Variance: 36, 16, 16, 64 for the categories A, E, G, and T, respectively.
- {a,e,g,t}DIMACSg9w: Representation: Full text. Weighting: TFxIDF with cosine normalization. Prior: Gaussian. Hyperparameter Variance: 49, 25, 9, 64 for the categories A, E, G, and T, respectively.
- {a,e,g,t}DIMACSl9w: Representation: Full text. Weighting: TFxIDF with cosine normalization. Prior: Laplace. Hyperparameter Variance: 49, 49, 16, 100 for the categories A, E, G, and T, respectively.

All of the submitted runs used MEE thresholding (see Section 2.2 for the thresholds used on a probability scale). All runs used full 5-fold cross-validation on the training set to choose a hyperparameter (shown above for each run) from the values listed in Section 2.1.

The combinations of techniques submitted were chosen by cross-validation experiments on the training data. Not all combinations were exhaustively tried.

4.1 Official Results

Our official results, along with NIST-supplied statistics, are summarized in Table 4. The detailed results of our official runs are shown in Table 5.

Our run tDIMACSg9w achieved the best score for the "tumor" task among all submitted runs. We obtained very good results (very close to the best scores) on the "allele" and "expression" tasks, and slightly above median results on the "GO" task.

Our results indicated that full text representation was better than MEDLINE representation. With full text representation, the results with Laplace and Gaussian priors were very close. Gaussian priors usually gave better precision than Laplace priors, but worse recall. MEE thresholding was considerably more effective than TROT thresholding, which suggests a benefit to this approach when the desired tradeoff between false positives and false negatives is extreme.

[3]http://eutils.ncbi.nlm.nih.gov/entrez/query.fcgi
[4]http://www-2.cs.cmu.edu/∼lemur
[5]ftp://ftp.cs.cornell.edu/pub/smart/english.stop or http://jmlr.csail.mit.edu/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm

TASK	Our official runs				Statistics of submissions			
	g9md	l9md	g9w	l9w	Best	Median	Worst	No of Runs
allele	0.8221	0.8212	0.8168	**0.8292**	0.8710	0.7785	0.2009	48
expression	0.6720	0.6278	0.7976	**0.8491**	0.8711	0.6548	-0.007	46
GO	0.4603	0.4700	0.4538	**0.4809**	0.5870	0.4575	-0.034	47
tumor	0.9264	0.8268	**0.9433**	0.9069	0.9433	0.7610	0.0413	51

Table 4: Comparison of our official results and NIST-supplied statistics on effectiveness of official categorization task submissions in terms of normalized utility scores. Our official runs are shown with their run suffixes.

Category: A (allele)										
Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
aDIMACSg9md	294	358	38	5353	0.4509	0.8855	0.5976	4640	5644	0.8221
aDIMACSl9md	301	482	31	5229	0.3844	0.9066	0.5399	4635	5644	0.8212
aDIMACSg9w	289	303	43	5408	0.4882	0.8705	0.6255	4610	5644	0.8168
aDIMACSl9w	298	386	34	5325	0.4357	0.8976	0.5866	4680	5644	0.8292

Category: E (expression)										
Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
eDIMACSg9md	77	412	28	5526	0.1575	0.7333	0.2593	4516	6720	0.6720
eDIMACSl9md	76	645	29	5293	0.1054	0.7238	0.1840	4219	6720	0.6278
eDIMACSg9w	88	272	17	5666	0.2444	0.8381	0.3785	5360	6720	0.7976
eDIMACSl9w	95	374	10	5564	0.2026	0.9048	0.3310	5706	6720	0.8491

Category: G (GO)										
Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
gDIMACSg9md	326	963	192	4562	0.2529	0.6293	0.3608	2623	5698	0.4603
gDIMACSl9md	340	1062	178	4463	0.2425	0.6564	0.3542	2678	5698	0.4700
gDIMACSg9w	309	813	209	4712	0.2754	0.5965	0.3768	2586	5698	0.4538
gDIMACSl9w	346	1066	172	4459	0.2450	0.6680	0.3585	2740	5698	0.4809

Category: T (tumor)										
Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
tDIMACSg9md	20	340	0	5683	0.0556	1.0000	0.1053	4280	4620	0.9264
tDIMACSl9md	19	569	1	5454	0.0323	0.9500	0.0625	3820	4620	0.8268
tDIMACSg9w	20	262	0	5761	0.0709	1.0000	0.1325	4358	4620	0.9433
tDIMACSl9w	20	430	0	5593	0.0444	1.0000	0.0851	4190	4620	0.9069

Table 5: Details of our official text categorization task results for each category.

4.2 Two-Stage Classifiers

We reported the importance of the MeSH term "Mice" on the Gene Ontology category in our TREC 2004 genomics track experiments [1]. In addition to one-stage thresholded logistic regression models, we had also tested the following two-stage classifier on the triage task last year:

1. IF a document does NOT contain the MeSH term "Mice" classify it as negative.
2. ELSE classify it using a thresholded logistic regression model.

We did not have time to repeat this work for our official submissions this year. However, we followed this up after the submissions by training the logistic regression models only on training examples containing the MeSH term "Mice". Table 6 shows the post-submission runs corresponding to official runs but using two-stage classifiers.

The results show that the run configuration we used last year, which had given the best score then, still gives the best score this year, see Table 6. When we compare with our official results, two stage classifiers show improvement only for the "GO" category, but does not decrease the effectiveness much for the other categories. However, running two-stage classifiers are much faster.

Acknowledgements

The work was partially supported under funds provided by the KD-D group for a project at DIMACS on Monitoring Message Streams, funded through National Science Foundation grant EIA-0087022 to Rutgers University. The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency.

Category: A (allele)										
Corresponding Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
aDIMACSg9md	303	515	29	5196	0.3704	0.9127	0.5270	4636	5644	0.8214
aDIMACSl9md	314	677	18	5034	0.3169	0.9458	0.4747	4661	5644	0.8258
aDIMACSg9w	297	372	35	5339	0.4439	0.8946	0.5934	4677	5644	0.8287
aDIMACSl9w	298	417	34	5294	0.4168	0.8976	0.5692	4649	5644	0.8237

Category: E (expression)										
Corresponding Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
eDIMACSg9md	92	856	13	5082	0.0970	0.8762	0.1747	5032	6720	0.7488
eDIMACSl9md	86	1008	19	4930	0.0786	0.8190	0.1435	4496	6720	0.6690
eDIMACSg9w	94	428	11	5510	0.1801	0.8952	0.2998	5588	6720	0.8315
eDIMACSl9w	93	372	12	5566	0.2000	0.8857	0.3263	5580	6720	0.8304

Category: G (GO)										
Corresponding Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
gDIMACSg9md	408	1331	110	4194	0.2346	0.7876	0.3615	3157	5698	0.5541
gDIMACSl9md	449	1600	69	3925	0.2191	0.8668	0.3498	3339	5698	0.5860
gDIMACSg9w	410	1309	108	4216	0.2385	0.7915	0.3666	3201	5698	0.5618
gDIMACSl9w	442	1581	76	3944	0.2185	0.8533	0.3479	3281	5698	0.5758

Category: T (tumor)										
Corresponding Run	TP	FP	FN	TN	Precision	Recall	F1	Raw Utility	Max Utility	Norm Utility
tDIMACSg9md	20	724	0	5299	0.0269	1.0000	0.0524	3896	4620	0.8433
tDIMACSl9md	19	695	1	5328	0.0266	0.9500	0.0518	3694	4620	0.7996
tDIMACSg9w	20	348	0	5675	0.0543	1.0000	0.1031	4272	4620	0.9247
tDIMACSl9w	20	586	0	5437	0.0330	1.0000	0.0639	4034	4620	0.8732

Table 6: Results of post-submission runs corresponding to our official runs with two-stage classifiers.

5. REFERENCES

[1] A. Dayanik, D. Fradkin, A. Genkin, P. Kantor, D. Lewis, D. Madigan, and V. Menkov. DIMACS at the TREC 2004 genomics track. In *TREC '04*, 2005.

[2] Alexander Genkin, David D. Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. Technical report, DIMACS, 2004.

[3] W.R. Hersh, R.T. Bhuptiraju, L. Ross, A.M. Cohen, D.F. Kraemer, and P. Johnson. TREC 2004 genomics track overview. In *13th Text Retrieval Conference*, 2004.

[4] W.R. Hersh, A.M. Cohen, J. Yang, R.T. Bhuptiraju, P. Roberts, and M. Hearst. TREC 2005 genomics track overview. In *14th Text Retrieval Conference*, 2005.

[5] David D. Lewis. Evaluating and optimizing autonomous text classification systems. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *SIGIR '95: Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 246–254, New York, 1995. Association for Computing Machinery.

[6] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, July 1980.

[7] G. Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[8] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[9] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.