Experience of Using SVM for the Triage Task in TREC2004 Genomics Track

Dell Zhang[1,2]
[1]Department of Computer Science
School of Computing
S16-05-08, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2]Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744251
dell.z@ieee.org

Wee Sun Lee[1,2]
[1]Department of Computer Science
School of Computing
SOC1-05-26, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2]Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744526
leews@comp.nus.edu.sg

ABSTRACT

This paper reports our knowledge-ignorant machine learning approach to the triage task in TREC2004 genomics track, which is actually a text categorization problem. We applied Support Vector Machine (SVM) and found that information-gain based feature selection is helpful. Although we achieved decent performance in leave-one-out cross-validation experiments, the evaluation result on the test data turned out to be surprisingly poor. Further experiments revealed that there is a chasm between the training and test data distributions. It seems that more aggressive feature selection can partially alleviate the trouble caused by distribution change.

Keywords

Text Categorization, Machine Learning, Support Vector Machine, Feature Selection, Distribution Change.

1. INTRODUCTION

In this year's TREC conference, we have only tried to attack the triage task of the genomics track[1]. The goal of this task is to correctly identify which mouse-related papers have been deemed to have experimental evidence warranting annotation with GO[2] codes by MGI[3]. It is exactly a text categorization [11] problem. Since we do not have any biological or medical background, we just took a knowledge-ignorant machine learning [8] approach.

Support Vector Machine (SVM) [1, 10] is generally regarded as one of the most powerful machine learning methods. It has shown very promising performances in a number of recent text categorization studies [2, 6, 12]. An efficient SVM implementation, SVM*light*[4] [5], was used throughout our experiments.

In fact, we achieved decent performance in leave-one-out cross-validation experiments. However, to our surprise, the evaluation result on the test data is quite poor. We present our approach in section 2, and investigate the reason of failure in section 3.

2. OUR APPROACH

2.1 Data

The given document collection consists of mouse-related articles from three journals (JBC, JCB and PNAS) over two years (2002 and 2003). The first year's (2002) documents comprise the training data, while the second year's (2003) documents make up the test data. The training data include 375 positive examples and 5462 negative examples. The test data include 420 positive examples and 5623 negative examples.

Each example corresponds to a journal article which can be uniquely identified by its PMID. The SGML format full-text information of each example was provided. Furthermore, we fetched the XML format bibliographic information of each example from the PubMed[5] server.

We obeyed the "separate" strictness of data usage: no information from any test example is allowed to affect the processing of any other test example.

2.2 Features

To apply SVM, the data need to be represented as vectors. Inspired by the traditional bag-of-words [6] representation

[1] http://medir.ohsu.edu/~genomics/2004protocol.html

[2] http://www.geneontology.org/

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

[4] http://svmlight.joachims.org/

[5] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi

of text documents, we converted each example into a bag-of-features through the feature extraction and selection process explained later. Then we constructed a vector for each example based on its bag-of-features: the entries/dimensions of the vectors correspond to all distinct features, and the value of each entry is the weight of its corresponding feature. Here we used the SMART [9] word-vector-weighting[6] scheme *ltc*. Finally all vectors are normalized to have unit length.

For each example, the features are extracted from the following fields of its semi-structured bibliographic and full-text information: MESH, JOURNAL, CHEMICAL, GRANT, AUTHOR, AFFILIATION, TITLE, ABSTRACT, ST (section title) and CAPTION (table/figure caption). Especially the CAPTION fields have been reported to be quite useful for a similar task [14].

MESH [7] (Medical Subject Headings) is a controlled vocabulary produced by the National Library of Medicine and used for indexing, cataloging, and searching for biomedical and health-related information and documents. All MESH descriptor are organized in a tree structure. Each MESH descriptor can be mapped to a specific node in the MESH tree. For each MESH descriptor of the given example, we found its corresponding node in the MESH tree. Then we would generate a feature for every node in the path from the root to that node. If the MESH descriptor is modified by a qualifier (subheading), we would generate a new feature identified by that descriptor plus that qualifier. If the MESH descriptor/qualifier is considered describing the major topic of the document, we would generate another new feature indicating that it is a major MESH term. For example, the MESH term

```
<MeshHeading>
   <DescriptorName MajorTopicYN="N">
      Transcription Factors
   </DescriptorName>
   <QualifierName MajorTopicYN="Y">
      physiology
   </QualifierName>
</MeshHeading>
```

would be converted into a set of features:

```
MESH_D12,
MESH_D12_776,
MESH_D12_776_930,
MESH_D12_776_930_Q000502,
MESH_D12_776_930_Q000502_MAJOR,
```

where “`D12.776.930`” indicates the position of the MESH descriptor “`Transcription Factors`” in the MESH tree, and “`Q000502`” is the ID of the qualifier “`physiology`”.

For the JOURNAL, CHEMICAL, GRANT, AUTHOR and AFFILIATION fields, every specific entity would be treated as a feature. For example, such kind of features of the example with PMID 11677243 would include: `JOURNAL_0021_9258`, `CHEMICAL_Plasmids`, `GRANT_ID_HL_4518`, and `AUTHOR_WM_Canfield`, etc.

For the TITLE, ABSTRACT, ST and CAPTION fields, the contained texts would be extracted and canonicalized by the UMLS SPECIALIST lexical tool LuiNorm[8], then we would generate two features for every specific term in the texts: one is the term itself, the other is the term tagged by its occurring field. For example, such kind of features of the example with PMID 11677243 would include: `clone`, `TITLE_clone`, `mouse`, `ABSTRACT_mouse`, `rna`, `CAPTION_rna`, etc.

The feature selection criterion we used is the *information-gain*, since it has been shown to work well for various text categorization tasks [3, 13]. The decline of information-gain across features is very sharp, as shown in Figure 1.

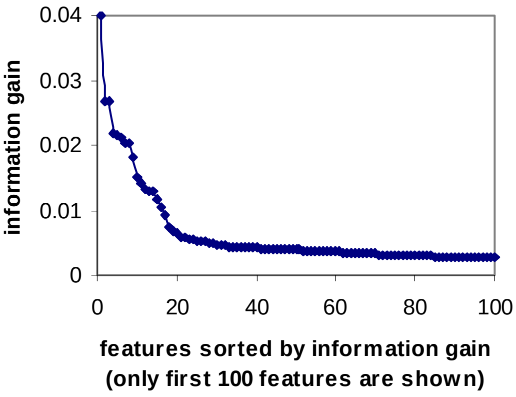

Figure 1: Distribution of features by information-gain.

It is generally believed that SVM requires no or very little feature selection for text categorization [6]. However, we have found that aggressive feature selection is very helpful to SVM for this specific problem. We think the reason is that a large number of features generated by the above feature extraction method are irrelevant or redundant. This finding is consistent with a recent study [3].

2.3 Runs

We used linear kernel and accepted the default values for all parameters of SVM*light* except *C* and *J*. The parameter *C* determines the trade-off between training error and margin, while the parameter *J* specifies the cost-factor by

[6] http://people.csail.mit.edu/people/jrennie/ecoc-svm/smart.html

[7] http://www.nlm.nih.gov/mesh/meshhome.html

[8] http://umlslex.nlm.nih.gov/lvg/2003/index.html

which training errors on positive examples outweight errors on negative examples. Another variable parameter is the feature selection threshold.

Our tactic for parameter tuning is pretty much like that of [7]. We trained SVM classifiers with different parameter settings and estimated their performance by leave-one-out cross-validation (LOOCV). SVM*light* can compute LOOCV performances very efficiently using a clever algorithm that prunes away cross-validation folds that do not need to be explicitly executed [4]. In addition, we found that a faster approximate version of pruning (the options "-x 1" and "-o 1") gave almost identical estimates as the exactly correct version of pruning (options "-x 1" and "-o 2"). A minor complexity was that SVM*light* only outputs LOOCV estimates of error-rate, precision and recall, but the official performance measure is the utility score defined as

$$U_{norm} = \frac{U_{raw}}{U_{max}} = \frac{u_r \bullet TP + u_{nr} \bullet FP}{u_r \bullet AP},$$

where $u_r = 20$ is the relative utility of a relevant document, and $u_{nr} = -1$ is the relative utility of a non-relevant document. The solution is to calculate the utility score based on the precision and the recall:

$$U_{norm} = recall + \frac{u_{nr}}{u_r} \bullet (\frac{1}{precision} - 1) \bullet recall$$

$$= recall - \frac{1}{20} \bullet (\frac{1}{precision} - 1) \bullet recall \, .$$

Our experiments showed that heuristically setting $C = 1 / 20$ and $J = 20 * (\#neg / \#pos)$ generated optimal LOOCV performances, where #pos and #neg represents the number of positive and negative training examples respectively. For the SVM with this parameter setting, the relationship between the LOOCV utility score and the number of selected features is shown in Figure 2. The optimal LOOCV performance was achieved using about 20,000 features, which is only 8% of all features.

We submitted four runs with slight different feature selection levels: nusbird2004a, nusbird2004b, nusbird2004d, and nusbird2004e. The best performing run among these four submissions is nusbird2004a that used 20,192 features. Its LOOCV utility score on the training data is 0.8892. However, its utility score on the test data is only 0.2302, even worse than the baseline run that classifies all examples as positive. Such a large discrepancy is surprising.

In addition to these four regular runs, we also submitted another run nusbird2004c, using extra positive training examples from the ground-truth MGI database. On the MGI website, there is a database report file named go_refs.mgi[9], which is supposed to contain all "references used in GO annotations to mouse Markers". There were 3,817 examples in the go_refs.mgi file (dated on Aug 28, 2004) after removing those contained in the training and test data. We took those examples as extra positive examples, and used them to augment the training data. The run nusbird2004c set the parameter $C = 1$, $J = 20 * (\#neg / \#pos)$, and used 9,162 features. Its LOOCV utility score on the training data is 0.8968, and its utility score on the test data is 0.4440. This is our best official evaluation result. Note that the extra positive training examples did not include any test example. A few extra positive training examples were published after 2002. We also did experiments with those "future" examples removed from the augmented training data and found that the performance was not affected.

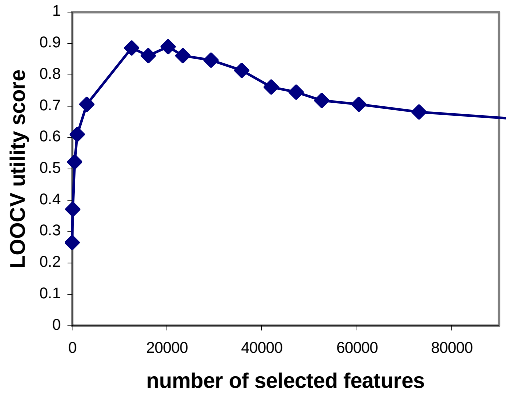

Figure 2: LOOCV performance on the training data at different feature selection levels.

3. DATA DISTRIBUTION CHANGE

Why did the classifiers with high LOOCV performance on the training data worked so poorly on the test data? We suspect that this strange phenomenon is due to the data distribution change. Most machine learning algorithms, including SVM, make the assumption that all training and test examples are independently and identically distributed. The data distribution change could make the basis of SVM invalid.

To scrutinize this problem, we model the training and test data as two multinomial probability distributions and employ KL-divergence (also known as relative entropy) to measure their dissimilarity. Note that this can only be done with the availability of the test instances although it does not require the test labels. At this stage, we only used the

[9] ftp://ftp.informatics.jax.org/pub/reports/index.html#go

official training data, and stuck to the SVM parameters *C* = 1 / 20 and *J* = 20 * (#neg / #pos).

One possible reason for the data distribution change is that we have chosen many inappropriate features. Therefore we tried to apply more aggressive feature selection to improve the performance of SVM. With the feature set used by nusbird2004a, the KL-divergence between the training and test data distributions was 0.0236. We first unconditionally removed all features that occurred less than 3 times in the training data. The KL-divergence dropped to 0.0136. Meanwhile, the utility score on the test data dramatically increased from 0.2302 to 0.4935, although the LOOCV performance became worse. Then we performed more aggressive feature selection based on information-gain. The KL-divergence decreased while fewer features were selected, as shown in Figure 3. If 1,000 features were selected, the KL-divergence would be 0.0121 and the utility score on the test data would be 0.5779.

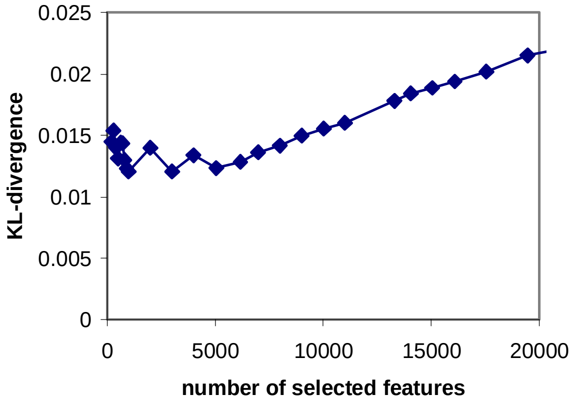

Figure 3: KL-divergence between the training and test data distributions at different feature selection levels.

Another factor that could be partially responsible for the data distribution change is the labeling error (noise). We compared the training data with the ground-truth positive data, go_refs.mgi (dated on Aug 8, 2004). There were 233 examples in their intersection, while 52 of them were labeled negative, i.e., mis-labeled. Therefore the positive noise rate (the fraction of mis-labeled positive examples among all positive examples) was roughly at the scale of 22.32% (52/233). Such a high level positive noise rate was not negligible, especially when using a utility score dominated by recall to measure performance.

4. CONCLUSION

This triage task is simply a binary text categorization problem, yet it has some interesting properties: the documents are semi-structured, the evaluation measure is a utility score that puts very high weights on true positive examples, the number of positive examples is small and the training and test data distributions have a noticeable difference. Our preliminary finding is that feature selection plays an important role to help SVM achieve good classification performance for this task. Its effect is twofold: removing irrelevant/redundant features and coping with distribution change.

Our best submitted run got the utility score 0.4440, with the help from information-gain based feature selection and extra positive training examples. Through more aggressive feature selection, we can achieve the utility score as high as 0.5779. How far can we expect a knowledge-ignorant approach to go for this task?

5. REFERENCES

[1] Cristianini, N. and Shawe-Taylor, J. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, 2000.

[2] Dumais, S., Platt, J., Heckerman, D. and Sahami, M. Inductive Learning Algorithms and Representations for Text Categorization. in *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, Bethesda, MD, 1998, 148-155.

[3] Gabrilovich, E. and Markovitch, S. Text Categorization with Many Redundant Features: Using Aggressive Feature Selection to Make SVMs Competitive with C4.5. in *Proceedings of the 21st International Conference on Machine Learning (ICML)*, Banff, Alberta, Canada, 2004, 321-328.

[4] Joachims, T. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

[5] Joachims, T. Making large-Scale SVM Learning Practical. in Scholkopf, B., Burges, C.J.C. and Smola, A.J. eds. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press, Cambridge, MA, 1999.

[6] Joachims, T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. in *Proceedings of the 10th European Conference on Machine Learning (ECML)*, Chemnitz, Germany, 1998, 137-142.

[7] Lewis, D.D. Applying Support Vector Machines to the TREC-2001 Batch Filtering and Routing Tasks. in *Proceedings of the 10th Text Retrieval Conference (TREC)*, NIST, Gaithersburg, MD, 2001, 286-292.

[8] Mitchell, T. *Machine Learning*. McGraw Hill, New York, 1997.

[9] Salton, G. *The SMART Retrieval System*. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[10] Scholkopf, B. and Smola, A.J. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[11] Sebastiani, F. Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, *34* (1). 1-47.

[12] Yang, Y. and Liu, X. A Re-examination of Text Categorization Methods. in *Proceedings of the 22nd ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, Berkeley, CA, 1999, 42-49.

[13] Yang, Y. and Pedersen, J.O. A Comparative Study on Feature Selection in Text Categorization. in *Proceedings of the 14th International Conference on Machine Learning (ICML)*, Nashville, TN, 1997, 412-420.

[14] Yeh, A.S., Hirschman, L. and Morgan, A.A. Evaluation of Text Data Mining for Database Curation: Lessons Learned from the KDD Challenge Cup. *Bioinformatics*, *19* (Suppl. 1). i331-i339.