Large Scale Text Classification with Multi-label Naive Bayes

Antti PUURULA

(*Dept. of Computer Science, University of Waikato, Hamilton, New Zealand*)

Abstract- **Large scale multi-label text classification tasks are becoming common with online databases, such as Wikipedia and the DMOZ web directory. Current machine learning classifiers lack the scaling capabilities required to manage these databases. An efficient multi-label mixture model is proposed in this paper called Multi-label Naive Bayes (MLNB). The MLNB can be extremely efficiently trained with closed form and linear time estimation. An approximate inference algorithm called Extended Greedy Iterative Addition (EGIA) is proposed for sparse generative classifiers, using pruning techniques and exploiting data sparsity to reduce uhe practical time complexity of classification. The model and the inference algorithm are evaluated on the LSHTC2 datasets for large scale multi-label text classification, resulting in accuracy comparable to K-Nearest Neighbour, with sub-second classification times for multi-label classification of over 325000 labels and 1.6 million features.**

Key words-multi-label; text classification; mixture model; Wikipedia; naive Bayes; DMOZ; LSHTC2; MLNB; EGIA

Manuscript Number: 1674-8042(2011)suppl.-0038-08

doi: 10.3969/j.issn.1674-8042.2011.suppl.009

1 Introduction

The arrival of large scale online databases is providing new opportunities and challenges for machine learning. Databases such as Wikipedia, DMOZ, IMDB and Worldcat among others contain vast amounts of structured data for new types of machine learning applications. For Wikipedia alone the INEX2010 evaluation[1] had 9 categories of tasks, ranging from classification and clustering to finding relevant text snippets.

Both the scalability of classifiers and use of structured data are challenges for machine learning classification. In these tasks classifiers must efficiently scale to potentially millions of documents, terms and labels. With large scale data linear time scaling concerning each of these dimensions becomes a necessity.

Structured data poses the second major challenge. Most classification algorithms have been designed for the case of a vector input and a single variable output. With newer applications of machine learning a variety of variable types are encountered. Multi-label classification refers to the case where the output variable is a set of labels, or equivalently a binary vector of label occurrences.

Early research treated multi-label classification as a sequence of independent binary classification problems[2]. However, this Binary Relevance method ignores label correlations and results in low performance. Current high-performance solutions for multi-label text classification include Multi-label Decision Trees[3] and Ensembles of Classifier Chains[4]. Neither has linear time complexities for both training and classification required for scaling.

There seems to be a demand for models that can deal with both structured data and scaling. One promising approach is the use of finite mixture models[5-7]. These are generative models generalizing the well known Naive Bayes model[8,9], by using mixtures of components for label-conditional distributions of words. Mixture models also show potential for many other structured uses, one notable example being modelling of multi-field input in current ranking models[10].

In this paper a simple mixture model called Multi-label Naive Bayes (MLNB) is presented with linear time and closed form estimation. The estimation is done in a streaming framework with online model pruning. The algorithms for multi-label mixture models are improved by turning data sparsity into an advantage. Models parameters are stored in sparse hash tables, with linear interpolation to smoothing models. For classification using multi-label mixture models a new algorithm called Extended Greedy Iterative Addition (EGIA) is proposed. This reduces the time complexity of classification to

* **Received:** 2011-07-06
Corresponding author: Antti PUURULA

practically linear by exploiting data sparsity and using several pruning criteria.

Evaluation of the MLNB model is conducted on the Large Scale Hierarchical Text Classification Challenge 2 (LSHTC2)[11] evaluation datasets. These consist of 3 multi-label and partly non-hierarchical classification tasks using subsets of the DMOZ[12] web directory and English Wikipedia[13], with up to some hundreds of thousands of labels. While the full databases for these have cyclic category structures and contain from over half a million (English Wikipedia) to over a million (DMOZ) labels, the evaluation sets approach the complexity of real-world uses.

Section 2 of the paper provides an overview of the mixture model approach to multi-label text classification. Section 3 presents the MLNB model and the efficient EGIA classification algorithm. Section 4 describes experiments and current evaluation results on the LSHTC2 datasets. Section 5 completes the paper with a discussion on the model and further extensions of the mixture model approach for text classification tasks.

2 Multi-label mixture models for text classification

2.1 Naive Bayes text classification

Text classification in machine learning refers to the task of assigning a class to a text. One ubiquitous use of text classification is spam filtering, where an e-mail is given a binary label l, spam/not-spam. It is standard to represent the input text as a vector of word counts $\boldsymbol{W}=[w_1, \cdots, w_n]$, known as the bag-of-words representation. More complex cases involve structured texts and classes, such as multi-field inputs and multi-label outputs. Classification is viewed in general as a supervised problem, where classifier functions are trained from training data with known classes assigned for each input text. This can be contrasted with clustering of similar texts, which is mostly an unsupervised task.

Probabilistic approaches to classification use a statistical model to provide the classifier function. Most commonly the model structure is fixed and training consists of estimation of parameters. Generative classification models such as the Naive Bayes[8,9] attempt to model the joint distribution $p(\boldsymbol{W}, l)$ of word count vectors $\boldsymbol{W}$ and categorical label variables l. Classification for a given input word vector is then done by choosing the label that maximizes this joint distribution. In the following Naive Bayes and its mixture model generalizations are detailed.

The Multinomial Naive Bayes models the joint distribution as

$$p(\boldsymbol{W},l) = p(l)p_l(\boldsymbol{W}) \propto p(l)\prod_n p_l(n)^{w_n}, \quad (1)$$

where $p(l)$ is the prior probability of label l and $p_l(\boldsymbol{W})$ is the label conditional multinomial probability of word vector $\boldsymbol{W}$ given label l. In this paper the word index n is considered to be a sparse representation, spanning only the non-zero word counts.

The Multinomial Naive Bayes takes parameters for the label priors $p(l)$ and label conditional multinomials $p_l(\boldsymbol{W})$. Estimating Maximum Likelihood (ML) parameters from a training dataset of known $\boldsymbol{W}$ and l is trivial and highly efficient. For the label priors this involves counting the occurrence of each label and dividing the counts by the total sum of label counts. For each multinomial the probabilities are likewise estimated by summing and normalizing.

2.2 Mixture models of text

Finite mixture models are a modelling approach with over a century of use in statistics, capable of modelling any distribution, given sufficient data for parameter estimates. Mixture models have recently found use in text classification as a means for generalizing the label-conditional distributions used in Naive Bayes, resulting in much richer generative classification models. Simple assumptions have been used to extend these models to multi-label classification[5-7].

The earliest use of mixture modelling in text classification was not in extending the multinomial, but in smoothing low-count word estimates to prevent overfitting. Due to the power law distribution of words, most possible words occur once or not at all for a given label, resulting in insufficient data for parameter estimates. Interpolation of a multinomial with a smoothing distribution such as the uniform can be conveniently formalized as a mixture model.

The most common words on the other hand occur very frequently. This results in underfitting, as the multinomial only uses a single parameter per word, even when ample data is available. Use of mixtures enables more detailed modelling of the label-conditional word distributions, resulting in extensions of Naive Bayes with mixtures of multinomials[14] as

$$p_l(\boldsymbol{W}) \propto \sum_m k_{lm} \prod_n p_{lm}(n)^{w_n}, \quad (2)$$

where the introduced $\boldsymbol{K}_l$ is a component weight vector summing to 1.

A mixture model of this type can be called a *document -clustering mixture*, as each mixture component klm now becomes a prototype document. Another common type [15] of mixture model for text takes the form:

$$p_l(\boldsymbol{W}) \propto \prod_n \left[\sum_m k_{lm} p_{lm}(n) \right]^{w_n}. \qquad (3)$$

Here the mixture is over words, and model of this type can be called a *word-clustering mixture*. In practice these models work very differently. A variety of more complex mixture models have been designed for document representation over the past decade. Latent Dirichlet Allocation[16], for example, can be understood as a word clustering mixture using a Dirichlet distribution to generate k_{lm} for each document.

Addition of either type of mixture complicates estimation of the models. Unlike the labels, the introduced components k_{lm} are hidden variables for the purpose of estimation, making exact ML-estimation intractable. Expectation Maximization (EM)[17] is an iterative local search algorithm that is conventionally used to estimate mixture models. EM operates by alternating between an Expectation step and a Maximization step until a local maximum of the likelihood function is reached. With mixture models these steps correspond to computing the conditional probabilities of the mixture components given the data $p(\boldsymbol{K}|\boldsymbol{W}, l)$ (E-step), followed by ML-estimation of parameters by assuming that the expected probabilities are the true component weights (M-step). If the likelihood function is unimodal or the initial parameters are suitably chosen, EM gives the exact ML estimate. Most commonly likelihood functions are multimodal, and additional strategies such as multiple restarts are often used to get approximate ML estimates with EM.

2.3 Multi-label mixture models of text

A number of publications following[5] have proposed the use of mixture models for multi-label text classification[6,7]. These multi-label mixture models share a number of choices in their modelling assumptions and the used algorithms. One model that has seen further development is the Parametric Mixture Model (PMM)[6]

$$p(\boldsymbol{W},\boldsymbol{L}) \propto \prod_n \left[\sum_m \frac{l_m}{\sum_j l_j} p_m(n) \right]^{w_n}, \qquad (4)$$

where $\boldsymbol{L} = [l_1, \cdots, l_{|L|}]$ is a binary label representation of the label set, with each l_m a binary variable instead of the categorical l as in the single-label case.

The PMM, as most multi-label mixture models, extend the multinomial Naive Bayes to the multi-label case by decomposing the multinomials conditional on the label sets $p_L(n)$ into additive components $p_m(n)$ conditional on the individual labels lm. In contrasu to most models PMM makes two strong assumptions; that the label set priors $p(\boldsymbol{L})$ are uniform, so that they can be omitted from (4), and that the label components are uniform, so that $k_m = l_m / \sum_j l_j$. As an example of the second assumption, in a document with five labels each label would generate one fifth of the words, $l_m / \sum_j l_j = 1/5$. Dirichlet priors corresponding to adding one to each word count are used for smoothing $p_m(n)$. Due to the uniform component weights PMM likelihood is convex, and an EM-like iterative algorithm was provided for exact Maximum A Posteriori estimation of the parameters.

Training for the PMM is relatively efficient, since the labels are known in training. Classification on the other hand becomes an NP-complete problem, as the labels are unknown and $|\boldsymbol{L}|^2$ label combinations exist. Approximate inference is used with multi-label mixture models to overcome this. The most important approximation is here called Greedy Iterative Addition (GIA). With GIA labels are iteratively added by choosing the label maximizing the label set probability, until the label set probability does not increase. This approximation turns a $|\boldsymbol{L}|^2$ time complexity probmem into a practically $|\boldsymbol{L}|$ complexity problem, enabling very efficient linear time classification. Details and an extension of this approximation are given in the next section.

3 Multi-label naive Bayes

3.1 Model definition

The PMM is the most computationally efficient multi-label mixture model and has lead to a number of more complex models. However, for large databases the EM-like iterative algorithm is undesirable, as what is needed is models that can be efficiently trained from a single pass over the training data. In addition the lack of label set priors $p(\boldsymbol{L})$ in the PMM unnecessarily degrades classification performance.

In this paper a model taking these factors into account is presented, called Multi-label Naive Bayes (MLNB):

$$p(\boldsymbol{W},\boldsymbol{L}) \propto p(L) \prod_n \left[\sum_m \frac{l_m}{\sum_j l_j} p_m(n) \right]^{w_n}. \qquad (5)$$

The label set prior can take many forms, but factorization of the binary vector $p(\boldsymbol{L})$ is not straightforward. One option, used in Ref. [5], is to have a categorical distribution over the label sets. For smoothing purposes both the priors and the multinomials can be implemented as interpolation mixtures.

The interpolation-smoothed label set priors take the form:

$$P(\boldsymbol{L}) = a_1 p^u(\boldsymbol{L}) + a_2 p^s(\boldsymbol{L}) + a_3 U, \qquad (6)$$

where a are the interpolation mixture weights, $p^u(\boldsymbol{L})$ the unsmoothed label set categorical, $\boldsymbol{U}$ the uniform distribution and $p^s(\boldsymbol{L})$ a categorical over the label sets with parameters tied between label sets of the same number of labels. Thus $p^s(\boldsymbol{L})$ works as a histogram model of different label set sizes.

The multinomials are similarly smoothed using an interpolation mixture:

$$p_{lm}(n) = a_4 p^u_{lm}(n) + a_5 p^s_m(n) + a_6 \boldsymbol{U}, \qquad (7)$$

where $p^s_m(n)$ is again a smoothing distribution, this time an average of the multinomial probabilities, working as a background distribution.

Use of two smoothing distributions is similar to use of two-stage smoothing in information retrieval[18]. Smoothing with a uniform complements the very rare cases where even the smoothing model p^s is not accurately modelled. Estimation of the model can be done almost as efficiently as Naive Bayes, since the probability mass from each training document is split evenly between the labels. This complicates estimation time only by $E(\sum_m l_m)$, the average number of labels per document.

The use of interpolation smoothing also enables sparse modelling, a crucial requirement for large scale classifiers. Both p^u and p^s can be represented as hash tables of occurring entries, omitting the vast majority of entries that have values of 0. Information related to a specific multinomial does not need to be stored, and in an extreme case only a single count per multinomial needs to be added to the hash table.

The MLNB can be efficiently trained from a data stream using a single pass. The hash table representation means that the numbers of words, labels or training documents do not need to be known apriori, but new entry types can be added on-the-fly to the hash tables as they are encountered in streamed training data. To fit the models in finite memory, the multinomial hash table can be pruned on-line by periodically removing counts under a pruning threshold and removing the lowest counts to a maximum number of counts. A small discount multiplier is applied to the counts after each pruning to keep the hash table from saturating with high count entries and not accepting new entries.

3.2 Efficient inference

Multi-label mixture models use different approximations to reduce the $|\boldsymbol{L}|^2$-time complexity inherent in multi-label classification to closer to $|\boldsymbol{L}|$. Greedy iterative addition performs $|\boldsymbol{L}|$ in practice, but there are a couple of problems with it. Firstly the greedy search means that if a label is added to the label set, the error is irrecoverable. Second, with large scale classification even $|\boldsymbol{L}|$ is not enough, due to large evaluation sets where label sets of size $|\boldsymbol{L}| > 300\ 000$ are encountered. Therefore more refined approximate inference strategies are required.

GIA can be improved both in terms of efficiency and coverage of the search space. The use of a smoothing histogram distribution $p^s(\boldsymbol{L})$ for the label sets indirectly improves the GIA, as the algorithm is biased against wandering to very large label sets in classification. This results in closer to linear time complexity concerning the label set size $|\boldsymbol{L}|$. A couple of more substantial improvements are presented in the following, resulting in the algorithm called Extended Greedy Iterative Addition (EGIA).

The first and foremost improvement is to construct after training a *reverse hash table*, containing for each word the list of labels $r(w)$ that have probability assigned to that word. The reverse hash table can then be used to reduce the time complexity of classification, which would be otherwise be linearly dependent on the number of possible labels $|\boldsymbol{L}|$. For each classification an *evaluation list* of possible labels can be constructed, by consulting the reverse hash table for the lists of labels that can generate the input words. For multinomial models the time complexity of classification reduces from nonsparse $E(\sum_w |\boldsymbol{L}|)$ to sparse $E(\sum_w |r(w)|)$. The complexity of $E(\sum_w |r(w)|)$ also becomes the practical time complexity of classification with EGIA, since in practice EGIA iterates only a small number of iterations that is independent of $|\boldsymbol{L}|$ and $|r(w)|$.

The evaluation list can be further ordered by the number of multinomial counts matching each label, weighted by the corresponding TFIDF-weighted word counts. This gives an approximate ordered list of the most likely labels to have generated the word vector. An *iteration pruning threshold* can then be used to terminate an iteration of EGIA, if it is unlikely that remaining labels in the evaluation list provide a better candidate for addition. A mean log likelihood is maintained for each iteration, and if the mean falls lower than the threshold from the current maximum log likelihood, the iteration is terminated.

Rejection of labels from consideration by threshold has been another popular strategy with multi-label mixture models. This can be integrated into EGIA by permanently rejecting from the evaluation list labels not improving the probability. This is used in the EGIA, as it does not seem to considerably reduce the accuracy of classification. Alternatively a label rejection threshold could be raised by a preset value, to only reject labels degrading the result considerably.

In order to provide GIA some ability to recover from errors, an additional *removal* step can be used before the addition step. For each label in the current label set removing the label is attempted, and the label is removed and rejected from the evaluation list if this improves the label set probability. Since the probability is still required to increase, this step doesn't cause backtracking. In theory, the removal step could complicate the worst case time complexity due to recurring addition and removal of labels, but with real data this does not occur. The added processing from the removal step is in practice minimal, and can even improve classification times due to interaction with pruning.

Tab.1 Extended greedy iterative addition

```
1  label _ set= {}; ordered _ list= {};
2  for each word $w_n$:
3      for each label $l_m$ in reverse _ hash _ table[w]:
4          add _ count(ordered _ list, $l_m$)
5  while ordered _ list! ={}:
6      prob _ cache= cache _ probs(label _ set)
7      for each label $l_m$ in label _ set:
8          try _ label _ removal(label _ set, $l_m$, prob _ cache)
9      if (remove(label _ set, max _ label)) continue
10     for each label $l_m$ in ordered _ list:
11         try _ label _ addition(label _ set, $l_m$, prob _ cache)
12         if (label _ prob< max _ prob)
13             remove(ordered _ list, $l_m$)
14         if (mean _ prob< max _ probiter _ threshold)
15             break
16     if(max _ prob oldry _ prob) break
17     add(label _ set, max _ label)
18     remove(ordered _ list, max _ label)
19 return label _ set
```

One more practical improvement possibly used by earlier authors using GIA is *caching* to keep the current label set probabilities for each word. This provides an exact time complexity reduction, since for each candidate label lm only a weighted sum for each word is required, instead of repeating the full computation of the joint probability $p(\boldsymbol{W}, \boldsymbol{L})$ for the proposed $\boldsymbol{L}$.

Tab.1 shows a coarse pseudocode description of the EGIA algorithm. Lines 2~4 construct the ordered evaluation list. Lines 5~18 contain the main iteration loop. Line 6 caches the per-word mixture probabilities from the current label set, 7~9 try removal of labels and 10~18 addition of labels. The proposed algorithm can be used with most of the multi-label mixture models that have been developed. The next section will show test results from the LSHTC evaluation datasets using the MLNB model with the EGIA algorithm for classification.

4 Experiments

4.1 Experiment datasets

In recent years multi-label text classification evaluations have started to scale up to the challenges encountered in online databases. The ongoing LSHTC2 evaluation uses the largest current datasets in terms of label set sizes. The datasets have been extracted from the English Wikipedia and DMOZ databases, and scale up to hundreds of thousands of labels and millions of words and training documents. The current full databases contain over a million labels for DMOZ and over half a million for English Wikipedia. In addition the databases contain cyclic category structures, which have been simplified to acyclic hierarchies for LSHTC2. The evaluation tasks nevertheless come very close to the actual real world task complexities.

Tab.2 shows the official statistics from the LSHTC2 datasets. The documents are provided in a sparse form with integer codes for labels and words, with the actual label and word identities undisclosed to the participants. The integer-coded word counts come from word stemming used by the LSHTC2 organizers. Label hierarchy descriptions are also provided, but with the MLNB the label hierarchies are not utilized.

Tab.2 LSHTC2 training dataset statistics

	# labels	# words	# train docs	$E(\|L\|)$
DMOZ	27 875	594 158	394 758	1.023
W. Small	36 504	346 299	456 886	1.859
W. Large	325 058	1 617 899	2 365 436	3.261

4.2 Experimental setup

The multinomial model of text is known to have several incorrect assumptions undermining its performance. With Naive Bayes models a couple of common feature pre-processing methods are used to overcome these flaws. The word counts were processed using length normalization followed by the common Term Frequency-Inverse Document Frequency (TF-IDF) weighting[9]:

$$w_n = \log\left[\frac{w_n^u}{\sum_r w_r^u} + 1\right]\log\frac{|D|}{d_n}, \qquad (8)$$

where w^u are the unprocessed word counts, $|D|$ the number of training documents and d_n the number of training documents containing wn. Since stream training was used, $|D|$ is not known beforehand. Therefore an adaptive Online TF-IDF version was used, so that $|D|$ was updated during the training with prior additive smoothing for the d_n. This allows the use of Online TF-IDF word weights in pruning the multinomial hash table during training.

A number of parameters had to be optimized on held-out test sets, namely the interpolation weights in the smoothing mixtures and the different pruning criteria used in training and classification. The training data sets for each 3 LSHTC2 tasks were split into dry-run development partitions dry train, dry dev and dry eval, with 1 000 documents for development set, 2 000 for the evaluation set and the rest for the training set.

Having a small development set enabled faster calibration of the parameters. A simple Random Optimization[19] search was run to set the meta-parameters, using the dry-train and dry-dev partitions. This is an iterative hill-climbing search, sampling normally-distributed points around the current maximum. The training was done on a computing cluster using batches, with variance of the sampling decreasing in linear step sizes. For training the optimized meta-parameters were for frequency of hash table pruning, count-based pruning threshold, maximum hash table size and additive smoothing for TF-IDF. For evaluation the meta-parameters were the interpolation smoothing weights and the iterative pruning threshold for EGIA. The final training times were very low for all datasets, the optimized Large Wikipedia model training took 34 minutes using Java on a single cluster machine with 4 GB of memory. For Small Wikipedia training took only 9 minutes and for DMOZ 6 minutes. A maximum of 8 million multinomial counts were stored in the hash tables, along 1.4 million label set counts for the Large Wikipedia task. Fig. 1 and Fig. 2 show the multinomial hash table and the reverse hash table for the Large Wikipedia task models. In both cases a power-law distribution is seen, as the most common words and labels are modelled with more counts by the sparse multinomials.

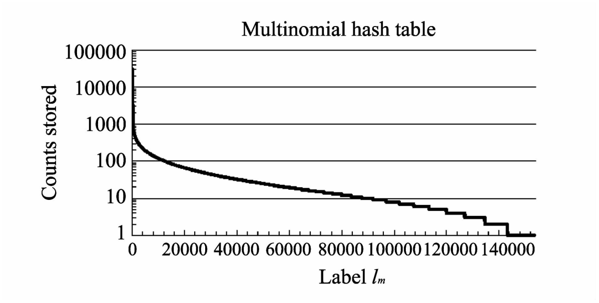

Fig. 1 Sorted multinomial hash table counts for the Large Wikipedia model. Counts were stored for 153 794 different labels, with a maximum of 31 309 words per label

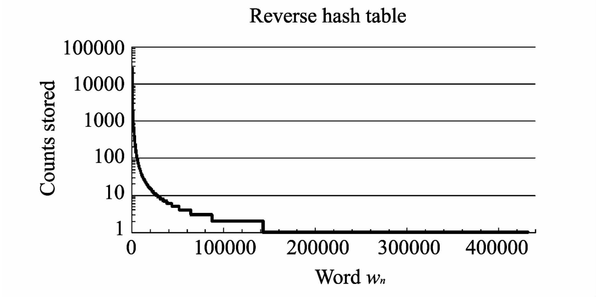

Fig. 2 Sorted reverse hash table counts for the Large Wikipedia model. Counts were stored for 431 821 different words, with a maximum of 27 036 labels per word

4.3 Results

Classification was performed on the dry_dev, dry_eval and the official evaluation test sets. For brevity only the label-based micro F-score (LBMiF) is reported. Among the many metrics for evaluation of multi-label text classification the LBMiF is the most commonly used. Tab. 3 shows the F-score results from the test sets in and the baseline K-Nearest Neighbours (KNN) results from LSHTC2 organizers. On the DMOZ data MLNB clearly outperforms the KNN baseline, 0.23 vs. 0.10. On the Wikipedia evaluation KNN gives much better results, 0.19 vs. 0.29 for Small and 0.16 vs. 0.30 for Large. The high variance in the results likely comes from either or both models not being accurately fitted to the tasks. For reference, at the time of writing the LSHTC2 evaluation is ongoing with daily updated results, but already some of the proposed classifiers are outperforming the KNN baseline across the tasks with F-score results ranging roughly from 0.28 to 0.38.

It can be concluded that the MLNB provides comparable baseline results to the KNN in these tasks, but without the very high computational cost of KNN of comparing test documents to every training document. The MLNB models used only roughly 0.4 seconds per classified document, or 11.0 minutes for DMOZ, 10.9 for Small Wikipedia and 14.0

for Large Wikipedia dry _ eval partitions of 2 000 documents. The KNN time complexity is linear to the number of training documents, and with larger databases this is bound to become inefficient.

Tab.3 LSHTC2 results in label-based micro F-score

	dry _ dev MLNB	dry _ eval MLNB	eval MLNB	eval KNN
DMOZ	0.2537	0.2372	0.2319	0.1074
W. Small	0.2112	0.2133	0.1953	0.2978
W. Large	0.1800	0.1862	0.1611	0.3015

5 Discussions

This paper presented a simple multi-label extension of the popular Multinomial Naive-Bayes model, evaluated in the ongoing LSHTC2 evaluation of large scale multi-label text classification. The Multi-label Naive Bayes (MLNB) mixture model has very efficient closed form estimation, resulting in training times of minutes on large datasets using a single computer. An efficient inference algorithm called the Extended Greedy Iterative Addition (EGIA) was proposed as a decoding strategy for MLNB and related mixture models. This resulted in sub-second classification times for multi-label classification with hundreds of thousands of labels and over a million features, with accuracies comparable to KNN baselines.

While the EGIA classification algorithm was developed for fast inference using sparse multi-label mixture models, it can be equally used in the single-label case by constraining the search to a single iteration. EGIA can therefore be used with sparse single-label Naive Bayes or any sparse classifier generalizing the Naive Bayes. Since large-scale classification problems use almost exclusively sparse or pruned models, EGIA can be used to scale generative models to very high dimensional classification without the normally associated costs.

It is clear from the evaluation tasks that many task-specific resources were not fully utilized, including the label hierarchies. In terms of classification accuracy it is obvious that more memory and time-consuming classifiers will outperform MLNB. The prime motivation of the research in this paper has not been the proposal of MLNB as an off-the-shelf solution for large multi-label text classification, but rather the development of a generative classification model for these tasks that can be used as a starting point for more complex mixture modelling. In this respect the research has been highly successful, as the MLNB provides baseline results with extremely low time and memory requirements.

The simple meta-heuristics search Random Optimization likely found less than optimal values for the Large Wikipedia, resulting in considerably higher training and classification times for that task. In current work search based on Simultaneous Perturbation Stochastic Approximation (SPSA)[20] is investigated. This should improve the MLNB in all accounts, as gains in model size, time requirements and accuracy are interrelated with the techniques used in this paper. For example, the use of the reverse hash table in EGIA means that the smaller the model gets pruned, the faster classification becomes. Efficient and reliable meta-optimization will also enable the addition of many more features that should improve the MLNB performance by a fair margin. These could include more models for the interpolation smoothing, log-linear weighting of the prior and the multinomial, pruning labels from the evaluation list by minimal TFIDF sum and more detailed feature preprocessing such as the BM-25.

Future research on generative classifiers for text classification can improve on the MLNB, while using many of the techniques presented in this paper. A number of increasingly complex mixture models have recently been designed for text classification[7]. Many of these however lack the scaling to large scale tasks that the MLNB possesses. A direction suited for generative modelling is addition of different document and word-level mixtures, while using a linear-time training algorithm such as Stepwise EM[21] for training the multi-layer mixture. In this way the extremely efficient properties of the MLNB could be coupled with the classification accuracy of the more complex generative classification models.

References

[1] De Vries C M, Nayak R, Kutty S, et al. Overview of the INEX 2010 XML mining track : clustering and classification of XML documents. Initiative for the Evaluation of XML Retrieval (INEX) 2010, Amsterdam, 2011.

[2] Tsoumakas G., Ioannis K. Multi-label classification: an overview. *International Journal of Data Warehousing and Mining*, 2007, 3(3): 1-13.

[3] Vens C, Struyf J, Schietgat L, et al. Decision trees for hierarchical multi-label classification. *Machine Learning*, 2008, 73: 185-214.

[4] Read J, Pfahringer B, Holmes G, et al. Classifier chains for multi-label classification. *Machine Learning and Knowledge Discovery in Databases*, *Lecture Notes in Computer Science*, *Springer Berlin / Heidelberg*, 2009, 5782: 254-269.

[5] McCallum A K. Multi-label text classification with a mixture model trained by EM. AAAI 99 Workshop on Text Learning, 1999.

[6] Ueda N, Saito K. Parametric mixture models for multi-labeled text. Advances in Neural Information Processing Systems, 2002, 15.

[7] Wang H, Huang M, Zhu X. A generative probabilistic model for multi-label classification. 8th IEEE International Conference on Data Mining, 2008: 628-637.

[8] Maron M E. Automatic indexing: an experimental inquiry. *Journal of the ACM*, 1961, 8:404-417.

[9] Rennie J D, Shih L, Teevan J, et al. Tackling the poor assumptions of naive bayes text classifiers. Proceedings of the Twentieth International Conference on Machine Learning, 2003:616-623.

[10] Wang K, Li X, Gao J. Multi-style language model for web scale information retrieval. Proceedings of SIGIR, SIGIR'10, 2010: 467-474.

[11] http://lshtc. iit. demokritos. gr.

[12] http://www. dmoz. org.

[13] http:/en. wikipedia. org.

[14] Nigam K, McCallum A K, Thrun S, et al. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 2000, 39: 104-134.

[15] Li H, Yamanishi K. Document classification using a finite mixture mode. Proceedings of ACL, Spain, 1997: 39-47

[16] Blei D, Ng A, Jordan M I Latent dirichlet allocation. *J. Mach. Learn. Res.*, 2003, 3: 993-1022 .

[17] Neal R M, Hinton G E. A view of the EM algorithm that justifies incremental, sparse, and other variants. Learning in graphical models, MIT Press, 1999: 355-368.

[18] Zhai C, Lafferty J. Two-stage language models forinformation retrieval. SIGIR 2002: 49-56.

[19] Matyas J. Random optimization. *Automation and Remote Cntrol*, 1965, 26: 246-253.

[20] Spall J C. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Trans. on Automatic Control*, 1992, 37: 332-341.

[21] Sato M, Ishii S. On-line EM Algorithm for the Normalized Gaussian Network. *Neural Comput.*, 2000, 12: 407-432.