Integrated Instance- and Class-based Generative Modeling for Text Classification

Antti Puurula
The University of Waikato
Private Bag 3105
Hamilton 3240, New Zealand
asp12@students.waikato.ac.nz

Sung-Hyon Myaeng
KAIST
335 Gwahangro
Daejeon 305-701, South Korea
myaeng@kaist.ac.kr

ABSTRACT

Statistical methods for text classification are predominantly based on the paradigm of class-based learning that associates class variables with features, discarding the instances of data after model training. This results in efficient models, but neglects the fine-grained information present in individual documents. Instance-based learning uses this information, but suffers from data sparsity with text data. In this paper, we propose a generative model called Tied Document Mixture (TDM) for extending Multinomial Naive Bayes (MNB) with mixtures of hierarchically smoothed models for documents. Alternatively, TDM can be viewed as a Kernel Density Classifier using class-smoothed Multinomial kernels. TDM is evaluated for classification accuracy on 14 different datasets for multi-label, multi-class and binary-class text classification tasks and compared to instance- and class-based learning baselines. The comparisons to MNB demonstrate a substantial improvement in accuracy as a function of available training documents per class, ranging up to average error reductions of over 26% in sentiment classification and 65% in spam classification. On average TDM is as accurate as the best discriminative classifiers, but retains the linear time complexities of instance-based learning methods, with exact algorithms for both model estimation and inference.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Statistical computing; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*

Keywords

Text Classification, Multinomial Naive Bayes, Generative Modeling, Tied Document Mixture, Instance-based Learning, Sparse Posterior Inference

1. INTRODUCTION

ADCS '13, December 05 - 06 2013, Brisbane, QLD, Australia

Statistical methods for text classsification are mostly based on the paradigm of class-based learning, where learning algorithms associate parameter values with class-variables, discarding information related to individual instances of training data after parameter estimation. Popular models such as Multinomial Naive Bayes (MNB) [26], Logistic Regression (LR) and Support Vector Machines (SVM) [12] are all examples of this paradigm. Class-based learning has the benefit of efficiency, as maintaining information for each class instead of the original documents reduces the requirements for storage and computation in most applications. However, discarding instance-specific information can reduce the effectiveness of learning algorithms, as the fine-grained information present in individual documents is summarized by the class-associated parameters.

Instance-based learning is an alternative paradigm that uses individual training instances for inference. K-Nearest Neighbors (KNN) [7] and Kernel Density Classifiers (KDC) [23, 13] are the most widely applied instance-based learning methods. Instance- and class-based learning have also been combined to design hybrid methods that outperform both types of learning on their own [25, 5, 18]. Instance-based methods can make better local decisions in inference by using the information from the training documents most relevant for each test document. However, in general the use of individual documents increases the data sparsity problem, and basic applications of instance-based learning fail to reach the performance of class-based learning in text classification [4].

This paper presents a new generative Bayes model combining instance- and class-based learning called Tied Document Mixture (TDM). TDM replaces the class-conditional Multinomial distribution in MNB models by a mixture of Multinomials, with one hierarchically smoothed Multinomial component for each training document of the class. Alternatively, TDM can be seen as a Multinomial Kernel Density classifier that smooths the training documents by class centroids. By using parameter tying of the mixture components and sparse matrix representation for the parameters, TDM achieves exact algorithms for training and inference with the the same linear complexities as KNN and KDC classifiers. For posterior inference an algorithm is proposed with time complexity dependent on the sparsity of training documents. This results in a combined model with the advantages of MNB and KDC classifiers in scalability, while achieving accuracies competitive with optimized dis-

criminative classifiers.

2. TIED DOCUMENT MIXTURE

2.1 Bayes Models

Among the most popular methods for modeling text in data mining is the MNB model. As a generative model MNB can be used for different types of inference, including clustering, ranking and classification. In classification the advantage of MNB is trivial scaling to large scale tasks provided by exact and linear time algorithms for model estimation and inference. The disadvantage is lower effectiveness compared to computationally more demanding methods such as SVM. The lower accuracy is caused by the set of strong modeling assumptions that lead to the MNB model.

Using the terminology of graphical models, MNB is a generative Bayes model using Multinomial distributions for class-conditional distributions. A generative graphical model is a model of the joint probability distribution $p(\boldsymbol{w}, l)$ of features $\boldsymbol{w}$ and classes l. In text modeling feature variables commonly take the form of a word count vector $\boldsymbol{w} = [w_1, ..., w_N]$ describing the weights w_n of words n in a document. With Multinomial models the weights are further constrained to integer counts of words and commonly the word vector is represented sparsely, using vectors of indices and non-zero counts. The class variables commonly take the form of a categorical variable $l : 1 \leq l \leq L$ describing the category of the document, constrained to binary values for binary-class modeling and extended to binary vectors for multi-label models.

Practically all current generative models use the Bayes rule to factorize the joint distribution into two models with independent parameters $p(\boldsymbol{w}, l) = p(l)p(\boldsymbol{w}|l)$, where the first factor $p(l)$ is called the class prior and the second factor $p(\boldsymbol{w}|l)$ is called the conditional distribution. Naive Bayes models [19] make the second assumption that the class-conditional probabilities for different features are independent: $p(\boldsymbol{w}|l) = \prod_n p(w_n, n|l)$. Finally, MNB parameterizes these probabilities with a Multinomial, so that $p(\boldsymbol{w}|l) = Z(\boldsymbol{w}) \prod_n p_l(n)^{w_n}$, where $Z(\boldsymbol{w})$ is the normalizer: $Z(\boldsymbol{w}) = (\sum_n w_n)! / \prod_n w_n!$. The normalizer can be omitted in most uses, but to be precise it needs an additional term such as Poisson to generate document lengths, with the Multinomials for different lengths containing shared parameters. These three MNB assumptions result in estimation and inference with exact, closed form solutions and the required algorithms scale linearly or less in the numbers of features, documents and classes.

Much prior work has been done in correcting the MNB assumptions at the level of individual words, as can be done in some cases with n-grams [24] and TF-IDF transforms [26]. However, these corrections do not account for word co-occurrence information at the level of documents. Figure 1 illustrates this modeling error with MNB. When a class-conditional Multinomial is used, it is assumed that the instances of documents for a class are drawn from the same Multinomial. As shown in Figure 1, this assumption works roughly for correlated words given the class. But when the words are uncorrelated, a class-conditional Multinomial nevertheless predicts the words to co-occur. Therefore MNB predicts class-dependent correlations when none exist in the original instances. Likewise, real connections between words are diminished when word statistics from unrelated instances are pooled to form the class-conditional parameters. The model proposed in this paper can be viewed as an extension of MNB that shares its useful properties, but does not suffer from this modeling error.

2.2 Model Definition

Mixture modeling techniques have been introduced for several uses with Bayes models of text. The most common and earliest use is smoothing of class-conditional distributions to correct parameter estimates with sparse data [11]. A more recent use is replacing the class-conditionals with document mixtures [28, 22], in the Multinomial case taking the form: $p(\boldsymbol{w}|l) = Z \sum_{m \in M_l} c_{lm} \prod_{n=1}^{N} p_m(n)^{w_n}$, where M_l denotes the set of components $m : 1 \leq m \leq M$ corresponding to class l. The required mixture parameters for this consist of the number of components $|M_l|$, weights c_{lm} of each component in the mixture and the component models $p_m(n)$. Mixtures can be added at the level of classes, documents or words, leading to more complex models such as topic models [2]. However, the increased model complexity necessitates iterative algorithms with approximate solutions such as sampling and local optimization.

As an alternative to approximate solutions, for many text mining uses we would like to maintain the scalability and exact algorithms of MNB models. A common strategy to simplify models is parameter tying: constraining parameter values to be fixed to other values. The present paper proposes a multi-level mixture model using several different types of parameter tying. Contrary to many earlier mixture model extensions of MNB, it is shown to improve effectiveness while maintaining exact solutions and incurring a tolerable increase in required computation. We can start by replacing the class-conditional Multinomial in MNB by a uniform-weight mixture of Multinomials, with one Multinomial component $p_m(n)$ estimated for each training instance m of the class l. The joint distribution for TDM becomes:

$$p(\boldsymbol{w}, l) = p(l)p(\boldsymbol{w}|l) = p(l)\frac{Z(\boldsymbol{w})}{|M_l|} \sum_{m \in M_l} \prod_{n=1}^{N} p_m(n)^{w_n}, \quad (1)$$

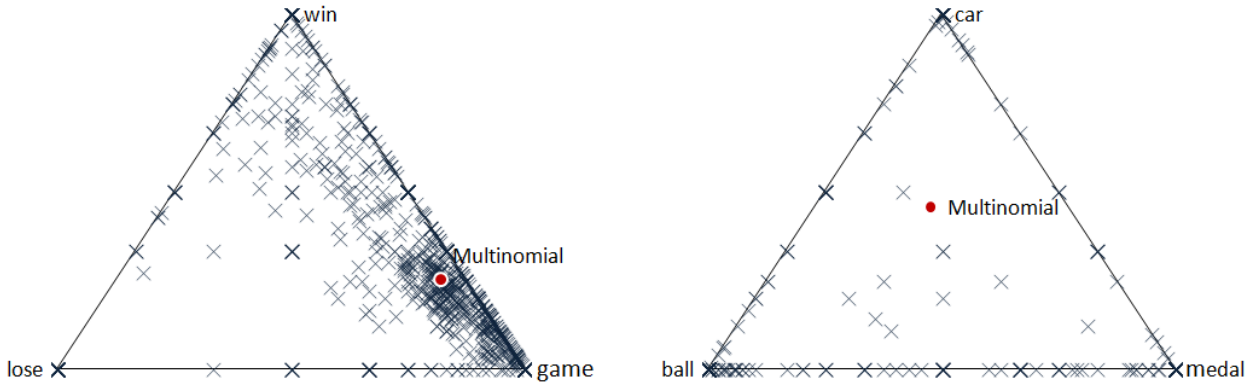
where $p(l)$ is Categorical and $Z(\boldsymbol{w})$ the Multinomial normalizer.

To reduce the data sparsity implicit in single instances, the instance Multinomials are smoothed hierarchically with a class-conditional Multinomial $p_l^s(n)$ and a Uniform background distribution $U(n)$:

$$p_m(n) = (1.0 - a_1 - a_2)p_m^u(n) + a_1 p_l^s(n) + a_2 U(n), \quad (2)$$

where $p_m^u(n)$ is the unsmoothed Multinomial for instance m and a_1 and a_2 are smoothing coefficients, so that $0 \leq a_1 \leq 1$, $0 \leq a_2 \leq 1$ and $a_1 + a_2 \leq 1$.

The class-conditional Multinomials are estimated by averaging the unsmoothed instance Multinomials:



(a) Correlated words.

(b) Uncorrelated words.

Figure 1: Parameters on a Multinomial 2-simplex for training instances (×) and a Multinomial (•) estimated from the instances. Instances are the first 1000 documents from English Wikipedia in "sports"-categories mentioning at least one of the three words forming the simplex. Left: for correlated words a Multinomial describes the center of distribution, but not the dispersion. Right: for uncorrelated words a single Multinomial describes neither the center nor dispersion of the document parameters.

$$p_l^s(n) = \frac{1}{|M_l|} \sum_{m \in M_l} p_m^u(n) \tag{3}$$

Finally, we can smooth the generative prior $p(l)$ by scaling and renormalizing $p(l) \propto p^u(l)^{a_3}$, where $a_3 \geq 0$ scales the influence of the unscaled prior $p^u(l)$. Scaling the prior, as well as background smoothing by $a_2 U(n)$, are commonly used for correcting generative modeling assumptions [11].

The TDM model combines tied mixture modeling of two different types: mixtures of instances at the document level and hierarchical smoothing at the word level. Figure 2 gives an illustration of these techniques. Use of uniform-weight mixtures of instances is closely related to Kernel Density Classifiers such as Flexible Naive Bayes [13]. Hierarchical smoothing is a common strategy in text modeling for text classification [28], information filtering [31] and retrieval [15]. For example, cluster-based document models for document retrieval [15] smooth document models hierarchically, but do not combine the conditional probabilities from documents for each cluster. Unlike earlier work in text modeling, TDM combines these two types of mixtures.

The parameter tying can be described as constraints on the two mixtures:

Document level mixture

- Number of components= $|M_l|$
- Components assigned to instances
- Component weights= $\frac{1}{|M_l|}$

Word level mixture

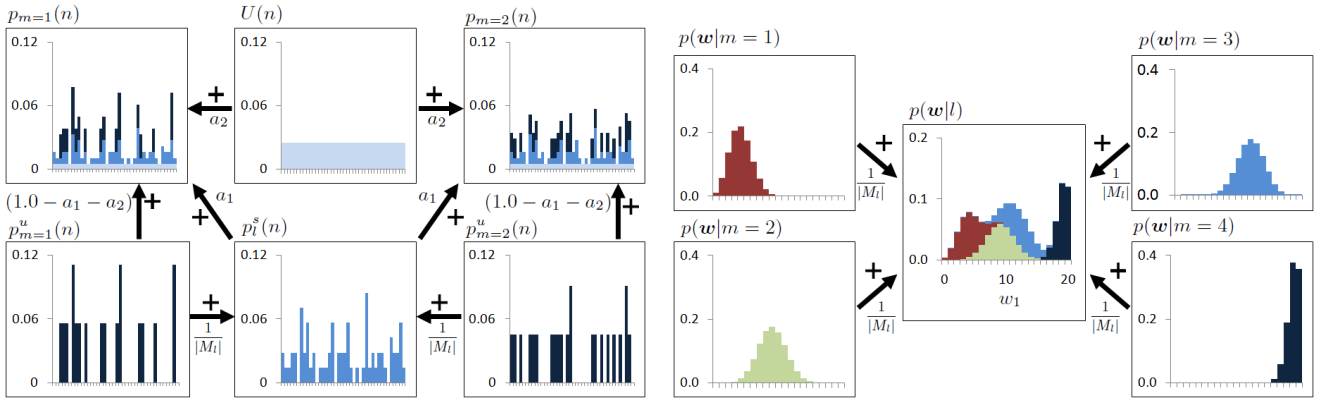
- Number of components= 3 (hierarchy depth)
- Components assigned to hierarchy
- Component weights= $1 - a_1 - a_2$, a_1 and a_2

Due to the parameter tying, estimating the model consists of simply normalizing instances by the document lengths $||\boldsymbol{w}||_1$, summing the instance statistics and normalizing to get $p_l^s(n)$ and $p(l)$, and finally smoothing the normalized parameters according to Equations 2 and 3. As will be shown, inference can likewise be done with a time complexity close to MNB, by utilizing sparse representations enabled by the parameter tying.

In the kernel density estimation framework [23, 13], TDM can be viewed as a Multinomial Kernel Density Classifier (KDC) that smoothes the training instances using class-centroids with the mean shift [10] method. In this view the document mixture $p(\boldsymbol{w}|l) = \frac{1}{|M_l|} \sum_{m \in M_l} p(\boldsymbol{w}|m)$ constitutes a Multinomial Kernel Density, where $p(\boldsymbol{w}|m)$ is a Multinomial Kernel function, shifted towards the class centroids by the hierarchical smoothing of Equation 2. However, the resulting kernel densities would be discrete, asymmetric, multivariate, bounded kernel functions, as well as local for each class. The kernel density literature treats each of these properties as a deviation to standard Parzen kernel densities and to our knowledge there is no prior work on using Multinomial or class-smoothed local kernel densities. The Multinomial KDC and KNN classifiers can be related to TDM by simplifying the model. By setting the centroid-based smoothing to $a_1 = 0$, TDM becomes a KDC with smoothed Multinomials as the kernels. By further approximating the summation in the kernel density with the top-k instances, the Multinomial KDC becomes a KNN with Multinomial distances and distance-weighted voting.

2.3 Sparse Representation and Inference

The model definition of Equations 1, 2 and 3 enables the use of sparsity for further improvements in efficiency. A naive implementation of TDM would use a dense representation in the form of vectors for representing the parameters. A corresponding naive inference algorithm for classification would result in a time complexity of $\Theta(||\boldsymbol{w}||_0 M)$, where $||\boldsymbol{w}||_0$ is the number of non-zero word counts in $\boldsymbol{w}$. We can reduce the complexities for both estimation and inference by taking note of parameter sparsity and the hierar-



(a) Hierarchical smoothing (b) Document mixture

Figure 2: Mixture modeling techniques combined in TDM. Left: Hierarchical smoothing is used to smooth the Multinomials for instances $p^u_{m=1}(n)$ and $p^u_{m=2}(n)$ with class Multinomial $p^s_m(n)$ and background Uniform $U(n)$, with interpolation smoothing coefficients a_1 and a_2, respectively. Right: class-conditional distributions are modeled with uniform mixtures of the smoothed training documents. Shown are probability mass functions for a Binomial case $|\boldsymbol{w}| = 2$, $|M_l| = 4$ and $w_1 + w_2 = 20$.

chical model structure. Instead of storing smoothed dense vectors, we can store the parameters $p^u_m(n)$ and $p^s_l(n)$ by using a sparse representation such as sparse vectors or a hash table. This reduces the complexities in estimation to $\Theta(e(||\boldsymbol{w}||_0)M)$, where $e(||\boldsymbol{w}||_0)$ is the average of non-zero words count per training instance. This is the same time complexity as MNB, KDC and KNN, and the same space complexity as the training instances.

More importantly, the inference time complexity can be reduced close to that of MNB by using suitable sparse representations. We can first precompute values, in order to improve efficiency and simplify equations. Instead of storing $p(l)$, $U(n)$, $p^s_l(n)$ and $p^u_m(n)$, we can store for non-zero parameters the precomputed values $p'(l) = \log(p(l)/|M_l|)$, $U'(n) = \log \frac{a_2}{N}$, $p'^s_l(n) = \log(\frac{a_2}{N} + a_1 p^s_l(n)) - U'(n)$ and $p'^u_m(n) = \log p_m(n) - p'^s_l(n) - U'(n)$. With these values, instance log-conditionals $\log p_m(\boldsymbol{w})$ decompose as $\log p_m(\boldsymbol{w}) = \sum_n w_n(U'(n)+p'^s_l(n)+p'^u_m(n))$, where $p'^s_l(n) = 0$ and $p'^u_m(n) = 0$ if the parameter is not stored. The values can be stored by inverted indices with postings lists $\boldsymbol{\zeta}_n$ for $p'^s_l(n)$ and $\boldsymbol{\eta}_n$ for $p'^u_m(n)$, where each posting in the lists consists of the pairs $(l, p'^s_l(n))$ for $\boldsymbol{\zeta}_n$ and $(m, p'^u_m(n))$ for $\boldsymbol{\eta}_n$. Using the inverted indices, the posterior distribution $p(l|\boldsymbol{w})$ can be computed efficiently from the instance log-conditionals.

Algorithm 1 shows this type of inference in the most basic form, returning the vector of posterior probabilities $p(l|\boldsymbol{w})$ for all classes l with the time complexity $O(M+\sum_{n:w_n\neq 0}(1+\sum_{m:p^u_m(n)\neq 0} 1))$, the same as for KNN [30]. The M term in the complexity can be further reduced by representing the tables *instance_scores*, *class_scores* and *m_retrieved* sparsely, and the algorithm can be further improved by techniques such as tree-based searches and maximum score bounding.

3. EXPERIMENTS

Table 1: Dataset statistics

Dataset	N	L	D_{train}	D_{eval}
TREC06	797772	2	35039	2783
ECUE1	100000	2	9978	1000
ECUE2	161155	2	10865	1000
ACL-IMDB	89527	2	47000	3000
TripAdvisor12	76364	2	60298	10077
Amazon12	86914	2	267875	100556
R8	14575	8	2785	1396
R52	16145	52	5485	2189
WebKb	7287	4	6532	2568
20Ng	54580	20	11293	7528
Cade	157483	12	27322	13661
RCV1-v2	160281	19587	343117	8644
EUR-Lex	172928	14240	17381	1933
OHSU-TREC	290117	196415	197555	35890

3.1 Experiment Datasets

Experiments were conducted on 14 different text classification datasets, all downloadable for research use [6, 8, 17, 1, 4, 14, 20, 27]. The preprocessed datasets and scripts are available at http://sourceforge.net/projects/sgmweka/. In the order of increasing difficulty, the datasets consist of binary-class (6), multi-class (5) and multi-label (3) classification tasks, with the binary-class tasks divided into spam classification (3) and sentiment analysis (3). Table 1 gives a summary of the datasets in this order, with the number of unique words, number of classes, and the training and evaluation dataset sizes in instances. Minimal standardization was done, to keep the results comparable with previous work. The binary-class sets used the original word vectors with little preprocessing. The multi-class and multi-label datasets used the original stemming and short-word removal, except for OHSU-TREC that had these added. In addition, the Label Powerset method [3] was used to transform the multi-label problems into multi-class problems. Hence the number of classes in Table 1 is for the number of unique sets

Algorithm 1 Sparse Posterior Inference for TDM

$smooth\_logprob = 0$
for all l **do**
  $class\_scores[l] = 0$
  $m\_retrieved[l] = 0$
for all m **do**
  $instance\_scores[m] = 0$
for all $n : w_n \neq 0$ **do**
  $smooth\_logprob+ = U'\, w_n$ ▷ $\rightarrow U'(\boldsymbol{w})$
  for all $(l, p'^s_l(n)) \in \boldsymbol{\zeta}_n$ **do**
    $class\_scores[l]+ = p'^s_l(n)\, w_n$ ▷ $\rightarrow p'^s_l(\boldsymbol{w})$
  for all $(m, p'^u_m(n)) \in \boldsymbol{\eta}_n$ **do**
    $instance\_scores[m]+ = p'^u_m(n)\, w_n$ ▷ $\rightarrow p'^u_m(\boldsymbol{w})$
    $m\_retrieved[l_m]+ = 1$ ▷ l_m: Class l of doc. m
for all l **do** ▷ Correct for non-retrieved m
  if $|M_l| - m\_retrieved[l] = 0$ **then**
    $class\_scores[l] = -1000000$
  else
    $class\_scores[l] = \log(|M_l| - m\_retrieved[l]) + class\_scores[l]$
for all $m \in instance\_scores[m]$ **do**
  $class\_scores[l_m] = \log(\exp(class\_scores[l_m]) + \exp(instance\_score[m]))$
$normalizer = -1000000$
for all l **do**
  $class\_scores[l]+ = smooth\_logprob + p'(l)$
  $normalizer = \log(\exp(normalizer) + \exp(class\_scores[l]))$
for all l **do**
  $class\_scores[l]- = normalizer$
return $class\_scores$ ▷ Return the exact $\log p(l|\boldsymbol{w})$

of labels. 5-fold development sets were used for ECUE1, ECUE2 and the single-label datasets. The other datasets used a single held-out development set. The datasets Amazon12, RCV1-v2 and OHSU-TREC had the highest document frequency words pruned to fit the datasets into 8 million unique counts. Both OHSU-TREC and RCV1-v2 had the original evaluation and training sets swapped, as this provided an order of more training documents that the models could utilize. In the cases where no evaluation set was provided, it was separated by random sampling from the training set.

3.2 Experiment Setup

The experiments compared TDM to class- and instance-based classifiers: MNB, KNN, KDC, SVM and LR. MNB, KNN, TDM and KDC all used Uniform-smoothed Multinomials. KNN used weighted voting for the combination, using the smoothed Multinomial probabilities as the voting weights for the top-k instances. For LR and SVM, L2-regularized Logistic Regression and Support Vector Machines were used, without and with TF-IDF (LR+, SVM+), with primal L2-regularized L2-loss training of SVM. A basic type of TF-IDF was used as a feature transform for the discriminative classifiers, taking the form $w_n = \log[1 + w^u_n]/||\boldsymbol{w}^u||_0 \log[M/M(n)]$, where $\boldsymbol{w}^u$ is the original feature vector and $M(n)$ the number of instances where the word n occurs. The algorithms MNB, TDM, KNN, KDC were implemented using the SGMWeka toolkit [1] v. 1.44. For LR, LR+, SVM and SVM+ the Liblinear toolkit[2] v.18 [9] was used with Python preprocessing.

The performance measure used in the experiments was micro-averaged F-score (miFscore) [29], calculated as the F_1-score from the summed statistics over each label for true positives, false negatives and false positives. This equals Accuracy in the binary- and multi-class cases, and is commonly used for the evaluation of multi-label classification. In addition, the binary-class datasets had balanced evaluation sets, making Accuracy a suitable measure for these tasks as well. Parameters required by the classifiers were optimized for the mean of F_1-score on the held-out development sets. For TDM these consisted of the 3 smoothing parameters a_1, a_2 and a_3, all constrained between 0 and 1. For comparable results, MNB and KDC used the prior scaled by a_3 and the smoothing parameter a_2 for Multinomials. KNN used a_2 and the k-parameter constrained between 1 and 50. The discriminative classifiers used the C-parameter constrained between 0.01 and 10.0. The parameters were optimized for each dataset using a parallelized Gaussian random search [16] using 40 iterations of 20 sampled points.

The training complexities of the MNB, TDM, KNN and KDC classifiers are linear and equal the sparsity of the feature-instance matrix : $\Theta(e(||\boldsymbol{w}||_0)M)$. Optimization of the parameters adds a constant multiplier to the complexities, equal to the number of sampled points. The largest time use for training on a single i7-2600 processor was for OHSU-TREC, with 70 s for MNB and 190 s for the others. In contrast, the discriminative classifiers LR, LR+, SVM and SVM+ require iterative algorithms. On the binary- and multi-class datasets these used up to 936 s, for LR on Amazon12. In the multi-label datasets the number of classes was too large for Liblinear and other linear classifier toolkits to handle in practice. Although problem transformation methods can be used to make the multi-label tasks more feasible for discriminative learning, this is a broad topic and these comparisons are left for future work. In terms of classification times the instance-based classifiers required more time, up to the maximum of 226 ms per classification on OHSU-TREC for TDM, compared to 70 ms for MNB.

3.3 Experimental Results

Classification results on the evaluation sets are shown in Table 2 in % miFscore. Significance tests were conducted for between-dataset differences using paired one-sided t-tests, reported with significances 0.005 (‡) and 0.05 (†). TDM can be compared to MNB, KDC and KNN on all datasets and in almost all cases TDM outperforms these in classification accuracy. On average the difference between MNB and TDM is significant and over 2.5%‡ miFscore absolute. Both KDC and KNN are significantly worse than MNB and TDM, by over 2.8%† difference to MNB and 5.3%‡ to TDM. This suggests that without smoothing by class centroids, instance-based learning for text classification is substantially worse than baseline methods. A notable exception is spam classification, where KDC and KNN outperform MNB by over

[1] http://sourceforge.net/projects/sgmweka/
[2] http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Table 2: MiFscore % of the classifiers on all datasets

Dataset	MNB	TDM	KNN	KDC	LR	LR+	SVM	SVM+
TREC06	99.21	**99.71**	99.39	99.39	99.35	99.57	99.50	99.68
ECUE1	92.70	**97.50**	95.70	94.80	95.20	93.50	95.40	97.00
ECUE2	96.00	**98.70**	96.80	96.70	98.30	96.40	97.60	98.60
ACL-IMDB	82.53	88.23	71.60	71.60	90.30	90.27	89.93	**90.53**
TripAdvisor12	86.76	88.44	77.90	77.73	91.89	91.98	91.87	**92.10**
Amazon12	80.33	**86.81**	79.75	80.13	84.34	84.36	84.06	84.29
R8	96.62	97.26	94.43	94.24	97.03	97.03	97.12	**97.58**
R52	91.90	91.24	90.77	90.50	92.99	92.48	93.15	**95.17**
WebKb	83.95	85.82	76.65	76.29	**91.62**	90.26	91.26	91.12
20Ng	82.09	83.98	73.29	73.31	80.15	84.14	80.74	**85.22**
Cade	59.67	60.31	51.21	51.84	58.11	60.02	57.72	**60.39**
RCV1-v2	64.98	**73.24**	69.89	70.53	NA	NA	NA	NA
EUR-Lex	48.11	**48.72**	48.29	48.33	NA	NA	NA	NA
OHSU-TREC	**40.23**	**40.23**	**40.23**	**40.23**	NA	NA	NA	NA

1.1%, but still fall behind TDM by over 0.6%.

Analyzing the means of accuracy measures gives more weight to datasets where the baseline accuracies are weak and gains in absolute terms can be made easily. Further analysis can be made on the results of Table 2 by examining averages of Relative Error Reduction (RER) instead of averages of the performance measure. RER can be defined as $1 - ((f^{max} - f^{s_2})/(f^{max} - f^{s_1}))$, where f^{max} gives the maximum performance measure score, f^{s_1} the score for baseline classifier and f^{s_2} for the new classifier reducing the error $f^{max} - f^{s_1}$. The relative reduction takes into account the baseline result and gives a more comparable score across different datasets. By averaging for each task type, we get the average RERs for TDM compared to MNB of 65.6%‡ for spam, 26.1%† for sentiment, 6.9% for multi-class and 8.3% for multi-label classification.

A simple interpretation of the results would claim that the TDM error reductions are proportional to the number of instances per class. Figure 3 illustrates RER compared to M/L. Although there seems to be a logarithmic correlation ($R^2 = 0.2999$), there is likely a more complex relationship underlying the error reductions. The different tasks in Figure 3 seem to form linearly separable clusters, with the gains in spam classification clearly deviating from the general trend. One likely explanation is that the word co-occurrences captured by instance-based learning are important for categorizing spam text and this is precisely the type of task where the pooled co-occurrence statistics of MNB lead to a modeling error.

The discriminative classifier results can be compared to TDM on the binary-label and multi-class datasets. The SVM+ classifier using TF-IDF outperformed the other methods in 6 out 11 datasets and by a margin of 1.06% average Accuracy. Only the difference between SVM+ and TDM was nonsignificant. However, the results depend on the task, with TDM outperforming the discriminative classifiers in the spam classification datasets, while SVM+ outperforms TDM in the multi-class datasets. In terms of RER these differences are substantial, with average RERs of at least 13.4%† for TDM in spam classification and 20.4%† for SVM in multi-class classification. It can be concluded that there is no statistically significant difference between the discriminative classifiers and TDM on average, but these preliminary results are dataset-dependent and in some datasets and tasks TDM can outperform discriminative classifiers, and vice versa.

4. DISCUSSION

This paper presented a new generative model for text classification, merging instance-based and class-based learning. Unlike earlier generative models for text, TDM combines both top-down and bottom-up integration of information sources, by smoothing instance-conditional Multinomials hierarchically and modeling class-conditionals as mixtures of the smoothed instances. TDM can be viewed either as a MNB replacing the class-conditionals by mixtures of hierarchically smoothed document models, or as a KDC using hierarchical smoothing of Multinomial kernel densities. TDM improves substantially on the performance of MNB, while maintaining exact linear time algorithms for estimation and inference. This is achieved by constraining the two mixture modeling techniques used in TDM. As a result, both the estimation and inference time complexities used by the model were shown to be the same as for KDC and KNN classifiers.

The model was evaluated on 14 different text classification datasets against MNB, KDC, KNN, LR and SVM baselines. In terms of effectiveness, TDM substantially and significantly improved on the performance of MNB, with the largest gains in tasks with many instances per class. The average error reductions ranged up to over 26% in sentiment classification and 65% in spam classification datasets. TDM performed on par with the state-of-the-art discriminative classifiers on average, with no significant difference in mean accuracy. This was however task-dependent, as TDM excelled in the case of spam classification, and was outperformed by L2-regularized SVM with TFIDF kernel in the case of multi-class classification. Additionally, TDM could be used on datasets with very large numbers of classes, whereas the discriminative classifiers could not be used on these. In summary, TDM is competitive in accuracy with strong discriminative classifier baselines, but has the scalability of instance-based learning methods such as KNN and

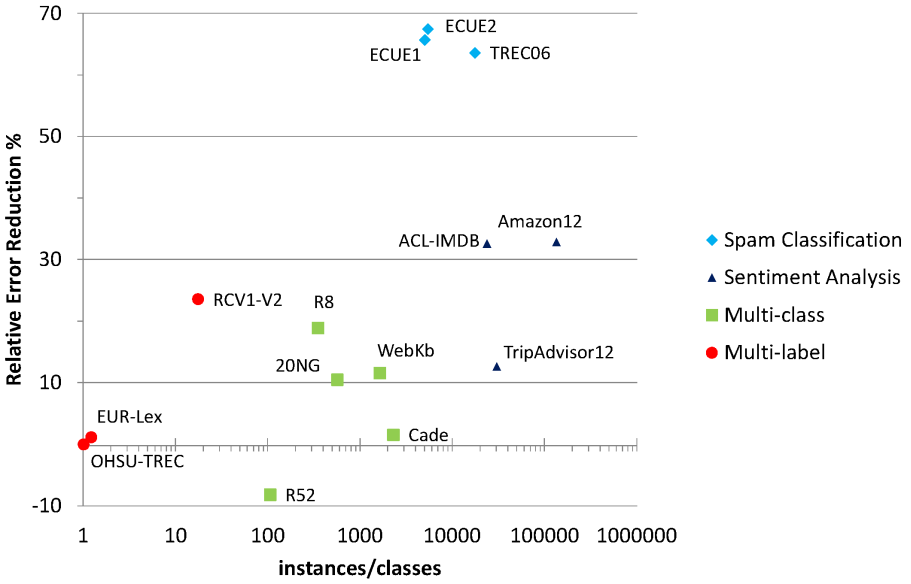


Figure 3: Relative Error Reductions of miFscore, comparing TDM to MNB

KDC.

The seamless combination of instance- and class-based learning makes numerous extensions to TDM possible, using techniques from different research traditions. Some of the most important ones can be noted in brief. 1) The hierarchical smoothing used in TDM can be extended into deep hierarchies. The time complexities will remain dependent on the sparsity of inputs and parameters, at worst linear in the number of nodes. Examples of nodes could be class and instance clusters, or any metadata features for the instances. 2) Integration of context-dependent models such n-grams models is possible. This would replace the hierarchical smoothing mixture in Equation 2 by a two-level mixture of the n-gram orders and the smoothing hierarchy. 3) The parameter tying can be relaxed, leading to tradeoffs between accuracy and scalability. Instead of fixed values, subsets of parameters can be optimized under a chosen loss function. For example, the instance weights in the document-level mixture could be optimized using the EM algorithm. 4) Further improvements in efficiency would be gained by integrating efficient search methods. These include tree-based searches common with KNN, query optimization strategies used with inverted indices in information retrieval, and decoder techniques used with large-scale language models.

The model and results presented in this paper are exciting for a number of reasons. Combining instance-based and class-based learning is a recent topic and there is little previous work in combining instance-based learning with generative class-based models. It was shown that by using parameter tying this combination can be done with efficient exact algorithms. It is commonly assumed that generative models are less effective on most tasks when sufficient data is available, based on classic results comparing the two paradigms [21]. This work showed that by merging instance-based learning it is possible to have generative models performing on par and possibly outperforming discriminative classifiers. The strong results on the binary-label tasks show that the integration of document-level information into statistical text models can lead to exceptional improvements in performance. This insight could be equally useful in many statistical models of text that currenly discard the document-level information available in large-scale text datasets.

4.1 Acknowledgments

This research was partially conducted during the first author's visit at Korea Advanced Institute of Science and Technology.

5. REFERENCES

[1] D. Bespalov, Y. Qi, B. Bai, and A. Shokoufandeh. Sentiment classification with supervised sequence embedding. In *ECML/PKDD (1)*, pages 159–174, 2012.

[2] D. M. Blei. Probabilistic topic models. *Commun. ACM*, 55(4):77–84, 2012.

[3] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757 – 1771, 2004.

[4] A. Cardoso-Cachopo. *Improving Methods for Single-label Text Categorization*. PhD thesis, Instituto Superior Técnico - Universidade Ténica de Lisboa, October 2007.

[5] W. Cheng and E. Hüllermeier. Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, 76(2-3):211–225, 2009.

[6] G. Cormack. TREC 2006 Spam Track Overview. In *Proceedings of TREC 2006*, 2006.

[7] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions*

on, 13(1):21–27, Jan. 1967.

[8] S. J. Delany, P. Cunningham, and B. Smyth. Ecue: A spam filter that uses machine learning to track concept drift. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29 – September 1, 2006, Riva del Garda, Italy*, pages 627–631, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

[9] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[10] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, Jan. 1975.

[11] F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

[12] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.

[13] G. H. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, UAI'95, pages 338–345, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[14] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.*, 5:361–397, Dec. 2004.

[15] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '04, pages 186–193, New York, NY, USA, 2004. ACM.

[16] S. Luke. *Essentials of Metaheuristics*. Lulu, version 1.2 edition, 2009. Available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/.

[17] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[18] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In D. N. Metaxas, L. Quan, A. Sanfeliu, and L. J. V. Gool, editors, *ICCV*, pages 89–96. IEEE, 2011.

[19] M. E. Maron. Automatic indexing: An experimental inquiry. *J. ACM*, 8:404–417, July 1961.

[20] E. L. Mencía and J. Fürnkranz. Efficient multilabel classification algorithms for large-scale problems in the legal domain. In E. Francesconi, S. Montemagni, W. Peters, and D. Tiscornia, editors, *Semantic Processing of Legal Texts – Where the Language of Law Meets the Law of Language*, volume 6036 of *Lecture Notes in Artificial Intelligence*, pages 192–215. Springer-Verlag, 1 edition, May 2010.

[21] A. Ng and M. Jordan. On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2001.

[22] J. Novovicova and A. Malik. Application of multinomial mixture model to text classification. In *Pattern Recognition and Image Analysis*, volume 2652 of *Lecture Notes in Computer Science*, pages 646–653. Springer Berlin / Heidelberg, 2003.

[23] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962.

[24] F. Peng and D. Schuurmans. Combining naive bayes and n-gram language models for text classification. In *In 25th European Conference on Information Retrieval Research (ECIR*, pages 335–350. Springer-Verlag, 2003.

[25] J. R. Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning, University of Massachusetts, Amherst, MA, USA, June 27-29, 1993*, pages 236–243, 1993.

[26] J. D. Rennie, L. Shih, J. Teevan, and D. R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *ICML'03*, pages 616–623, 2003.

[27] S. Robertson and D. A. Hull. The TREC-9 filtering track final report. In *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, pages 25–40, 2001.

[28] K. Toutanova, F. Chen, K. Popat, and T. Hofmann. Text classification in a hierarchical mixture model for small training sets. In *Proceedings of the tenth international conference on Information and knowledge management*, CIKM '01, pages 105–113, New York, NY, USA, 2001. ACM.

[29] G. Tsoumakas, I. Katakis, and I. P. Vlahavas. Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, pages 667–685. 2010.

[30] Y. Yang. Expert network: effective and efficient learning from human decisions in text categorization and retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 13–22, New York, NY, USA, 1994. Springer-Verlag New York, Inc.

[31] Y. Zhang, J. Callan, and T. Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 81–88, New York, NY, USA, 2002. ACM.